hw3

October 20, 2021

task1

[1]:
```
import pandas as pd
import numpy as np
import collections
```

[2]:
```
data = pd.read_csv('train',sep='\t',names = ['index','word','tag'])
```

[3]:
```
ttt = data.word.tolist()
```

[4]:
```
ttt_dict = collections.defaultdict(int)
for i in ttt:
    ttt_dict[i] = ttt_dict[i]+1
```

[5]:
```
temp_list = []
for w,o in ttt_dict.items():
    temp = [w,int(o)]
    temp_list.append(temp)
```

[6]:
```
def occrence(e):
    return e[1]
temp_list.sort(reverse=True, key = occrence)
```

[7]:
```
len(temp_list)
```

[7]:
```
43193
```

[8]:
```
big_list = []
low_frequency = set()
```

[9]:
```
i = 1
unknown = ['<unk>',0,0]
for w,o in temp_list:
    if o >= 2:
        temp = [w,i,o]
        big_list.append(temp)
        i +=1
    else:
        low_frequency.add(w)
        unknown[2] += o
```

[10]:
```
big_list.insert(0,unknown)
```

[11]:
```
len(big_list)
```

[11]:
```
23183
```

What is the selected threshold for unknown words replacement?

Answer:threshold for unknown words replacement is 4

What is the total size of your vocabulary

Answer:the size of my vocabulary is 13751

unknown number is 42044

[12]:
```
with open('vocab.txt','w') as f:
    for i in big_list:
        a = str(i[0]) + '\t' + str(i[1])+'\t'+str(i[2])+'\n'
        f.write(a)
```

[13]:
```
data_np = data.to_numpy()
```

[14]:
```
#creat a new list
new_list = []
for i in data_np:
    if i[1] in low_frequency:
        new_list.append([i[0],'<unk>',i[2]])
    else:
        new_list.append([i[0],i[1],i[2]])
```

[15]:
```
# convert the list into sentence
sentence_list = []

for i in range(len(new_list)):
    if new_list[i][0] == 1:
        temp = []
        temp.append(new_list[i])
    else:
        temp.append(new_list[i])
    if ((i+1) < len(new_list)) and new_list[i+1][0] == 1:
        sentence_list.append(temp)
```

[16]:
```
t_dict = collections.defaultdict(int)
e_dict = collections.defaultdict(int)
tag_dict = collections.defaultdict(int)
```

[17]:
```
for sentence in sentence_list:
    for i in range(len(sentence)):
        e_dict[sentence[i][2],sentence[i][1]] +=1
        tag_dict[sentence[i][2]] +=1
        if sentence[i][0] == 1:
            t_dict[('<s>',sentence[i][2])] += 1
        else:
            t_dict[(sentence[i-1][2],sentence[i][2])] += 1
```

[18]:
```
emission_dict = {}
```

[19]:
```
for key,value in e_dict.items():
    emission_dict[key] = value/tag_dict[key[0]]
```

[20]:
```
transitary_dict = {}
```

[21]:
```
sentence_num = len(sentence_list)
```

[22]:
```
for key,value in t_dict.items():
    if key[0] == '<s>':
        transitary_dict [key] = value/sentence_num
    else:
        transitary_dict [key] = value / tag_dict[key[0]]
```

[23]:
```
sentence_num
```

[23]:
```
38217
```

[24]:
```
json_transitary_dict = {}
for key,value in transitary_dict.items():
    json_transitary_dict[str(key)] = value
```

[25]:
```
json_emission_dict = {}
for key,value in emission_dict.items():
    json_emission_dict[str(key)] = value
```

[26]:
```
import json
```

[27]:
```
final_json = {"transition":json_transitary_dict,"emission":json_emission_dict}
```

[28]:
```
out_file = open("hmm.json", "w")
json.dump(final_json, out_file, indent = 6)

out_file.close()
```

0.1 hmm greedy

[29]:
```
#use sentence_list to appliment hmm greedy
```

[30]:
```
word_tag = collections.defaultdict(set)
```

[31]:
```
for sentence in sentence_list:
    for i in sentence:
        word_tag[i[1]].add(i[2])
```

[32]:
```
word_tag = dict(word_tag)
```

[33]:
```
def greedy_hmm(sentence):
    res = []
    for i in range(len(sentence)):
        target_word = sentence[i][1]
        probility_tag = []
        if target_word in word_tag:
            tag_list = list(word_tag[target_word])
        else:
            target_word = '<unk>'
            tag_list = list(word_tag['<unk>'])
        if sentence[i][0] == 1:
            for tag in tag_list:
                if ('<s>',tag) in transitary_dict:
                    t = transitary_dict[('<s>',tag)]
                else: t = 0

                if (tag,target_word) in emission_dict:
                    e = emission_dict[(tag,target_word)]
                else:
                    e = 0
                probility = t*e
                probility_tag.append(probility)

        else:
            for tag in tag_list:
                if (res[i-1],tag) in transitary_dict:
                    t = transitary_dict[(res[i-1],tag)]
                else: t = 0

                if (tag,target_word) in emission_dict:
                    e = emission_dict[(tag,target_word)]
                else:
                    e = 0
                probility = t*e
                probility_tag.append(probility)
        i_tag = tag_list[probility_tag.index(max(probility_tag))]
        res.append(i_tag)
    return res
```

[34]:
```
correct = 0
total = 0
for sentence in sentence_list:
    res_tag = greedy_hmm(sentence)
    for i in range(len(res_tag)):
        total += 1
        if res_tag[i] == sentence[i][2]:
            correct += 1
```

[35]:
```
correct/total
```

[35]:
```
0.9490553894497017
```

0.1.1 test greedy hmm

[36]:
```
test = pd.read_csv('dev',sep='\t',names = ['index','word','tag'])
```

[37]:
```
test_np = test.to_numpy()
```

[38]:
```
#creat a test list
new_test = []
for i in test_np:
    if i[1] in low_frequency:
        new_test.append([i[0],'<unk>',i[2]])
    else:
        new_test.append([i[0],i[1],i[2]])
```

[39]:
```
# convert the list into sentence
test_sentence_list = []

for i in range(len(new_test)):
    if new_test[i][0] == 1:
        temp = []
        temp.append(new_test[i])
    else:
        temp.append(new_test[i])
    if ((i+1) < len(new_test)) and new_test[i+1][0] == 1:
        test_sentence_list.append(temp)
```

[40]:
```
correct = 0
total = 0
for sentence in test_sentence_list:
    res_tag = greedy_hmm(sentence)
    for i in range(len(res_tag)):
        total += 1
        if res_tag[i] == sentence[i][2]:
            correct += 1
```

[41]:
```
accuracy_greedy_hmm = correct / total
```

[42]:
```
accuracy_greedy_hmm
```

[42]:
```
0.9352035278669611
```

0.1.2 produce greedy.out

[43]:
```
out = pd.read_csv('test',sep='\t',names = ['index','word'])
```

[44]:
```
out_np = out.to_numpy()
```

[45]:
```
#creat a test list
new_out = []
for i in out_np:
    if i[1] in low_frequency:
        new_out.append([i[0],'<unk>'])
    else:
        new_out.append([i[0],i[1]])
```

[46]:
```
# convert the list into sentence
out_sentence_list = []

for i in range(len(new_out)):
    if new_out[i][0] == 1:
        temp = []
        temp.append(new_out[i])
    else:
        temp.append(new_out[i])
    if ((i+1) < len(new_out)) and new_out[i+1][0] == 1:
        out_sentence_list.append(temp)
    if i == len(new_out)-1:
        out_sentence_list.append(temp)
```

[47]:
```
import copy

w_out_sentence_list = copy.deepcopy(out_sentence_list)

w_out_sentence_list

out_res = []
for sentence in out_sentence_list:
    res_tag = greedy_hmm(sentence)
    out_res.append(res_tag)

out_res

for i in range(len(out_res)):
    for i1 in range(len(out_res[i])):
        w_out_sentence_list[i][i1].append(out_res[i][i1])
```

[48]:
```
with open('greedy_out.txt','w') as f:
    for sentence in range(len(w_out_sentence_list)):
        if sentence != 0:
            f.write('\n')
        for (i, w, t) in w_out_sentence_list[sentence]:
            f.write(str(i))
            f.write('\t')
            f.write(str(w))
            f.write('\t')
            f.write(str(t))
            f.write('\n')
```

0.2 viterbi

[49]:
```
def viterbi_hmm(sentence):
    res = []
    for i in range(len(sentence)):
        target_word = sentence[i][1]
        probility_tag = {}
        if target_word in word_tag:
            tag_list = list(word_tag[target_word])

        else:
            target_word = '<unk>'
            tag_list = list(word_tag['<unk>'])
        if sentence[i][0] == 1:
            for tag in tag_list:
                if ('<s>',tag) in transitary_dict:
                    t = transitary_dict[('<s>',tag)]
                else: t = 0

                if (tag,target_word) in emission_dict:
                    e = emission_dict[(tag,target_word)]
                else:
                    e = 0
                probility = t*e
                probility_tag[tag] = ('<s>',probility)

        else:
            for tag in tag_list:
                previous_tag_list = []
                for previous_tag in res[i-1]:
                    if (previous_tag,tag) in transitary_dict:
                        t = transitary_dict[(previous_tag,tag)]
                    else: t = 0

                    if (tag,target_word) in emission_dict:
                        e = emission_dict[(tag,target_word)]
                    else:
                        e = 0
                    probility = t*e*res[-1][previous_tag][1]
                    previous_tag_list.append((previous_tag,probility))
                previous_tag_list = sorted(previous_tag_list,key = lambda x:
↪x[1],reverse = True)
                probility_tag[tag] = previous_tag_list[0]
        res.append(probility_tag)

    return res
```

[50]:
```
hhh = viterbi_hmm(sentence_list[77])
```

[51]:
```
def backtrace(table):
    tag_backtrace = []
    length = len(table)
    i = length -1
    end_col = table[i]
    end_tag = max(end_col, key=lambda key: end_col[key][1])
    tag_backtrace.append(end_tag)
    if i!=0:
        previous_tag = end_col[end_tag][0]
    i -= 1
    while i >= 0:
        tag_backtrace.append(previous_tag)
        previous_tag_col = table[i][previous_tag]
        i -= 1
        if i>= 0:
            previous_tag = previous_tag_col[0]
    tag_backtrace = list(reversed(tag_backtrace))
    return tag_backtrace
```

[52]:
```
table = viterbi_hmm(sentence_list[0])
```

[53]:
```
correct = 0
total = 0
for sentence in test_sentence_list:
    table = viterbi_hmm(sentence)
    res_tag = backtrace(table)
    for i in range(len(res_tag)):
        total += 1
        if res_tag[i] == sentence[i][2]:
            correct += 1
```

[54]:
```
hmm_viterbi_accuracy = correct/total
```

[55]:
```
viterbi_hmm(sentence_list[77][0:2])
```

[55]:
```
[{'VB': ('<s>', 1.2010925470418137e-07),
  'DT': ('<s>', 0.018196945407718525),
  'NNP': ('<s>', 7.906632093753297e-05)},
 {'RB': ('DT', 4.3783663451007866e-08),
  'IN': ('DT', 1.4395871855441722e-07),
  'JJ': ('DT', 3.323118608213109e-05)}]
```

hmm_viterbi_accuracy

[56]:
```
hmm_viterbi_accuracy
```

[56]:
```
0.9480231649095643
```

[57]:
```
w_out_sentence_list = copy.deepcopy(out_sentence_list)

w_out_sentence_list

out_res = []
for sentence in out_sentence_list:
    table = viterbi_hmm(sentence)
    res_tag = backtrace(table)
    out_res.append(res_tag)


for i in range(len(out_res)):
    for i1 in range(len(out_res[i])):
        w_out_sentence_list[i][i1].append(out_res[i][i1])
```

[58]:
```
with open('viterbi_out.txt','w') as f:
    for sentence in range(len(w_out_sentence_list)):
        if sentence != 0:
            f.write('\n')
        for (i, w, t) in w_out_sentence_list[sentence]:
            f.write(str(i))
            f.write('\t')
            f.write(str(w))
            f.write('\t')
            f.write(str(t))
            f.write('\n')
```

[59]:
```
print('The accuracy for greedy hmm is '+str(accuracy_greedy_hmm)+'\n The␣
 ↪accuracy for viterbi is '+str(hmm_viterbi_accuracy))
```

```
The accuracy for greedy hmm is 0.9352035278669611
 The accuracy for viterbi is 0.9480231649095643
```

[ ]: