```
import pandas as pd
import numpy as np
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
data_csv = pd.read_csv('/content/drive/MyDrive/hw2/data/train', on_bad_lines='skip', sep='\t', low_memory=False, names=["Index", "Word_T
data_csv
```

	Index	Word_Type	POS_Tag
0	1	Pierre	NNP
1	2	Vinken	NNP
2	3	,	,
3	4	61	CD
4	5	years	NNS
...	...	...	...
912090	22	to	TO
912091	23	San	NNP
912092	24	Francisco	NNP
912093	25	instead	RB
912094	26	.	.

912095 rows × 3 columns

```
data  = data_csv.groupby(["Word_Type"])["POS_Tag"].size().reset_index(name="Counts")
data = data.sort_values(by=['Counts'], ascending=False)
data
```

	Word_Type	Counts
32	,	46476
40681	the	39533
36	.	37452
33392	of	22104
40972	to	21305
...	...	...
6691	Brauchli	1
23850	countervailing	1
23849	countertop	1
23846	countersuing	1
26620	exerted	1

43193 rows × 2 columns

```
data['Word_Type'].mask(data['Counts'] <=3 ,'<unk>', inplace=True)
```

```
data
```

	Word_Type	Counts
32	,	46476
40681	the	39533
36	.	37452
33392	of	22104

```
# new_df =  data[['Word_Type', 'Counts']].copy()
# new_df
```

6691	<unk>	1

```
new_row = pd.DataFrame({'Word_Type' : '<unk>', 'Counts' : data['Word_Type'].value_counts()['<unk>']}, index=[0])
```

23849	<unk>	1

```
new_row
```

	Word_Type	Counts
0	<unk>	29443

```
data = data[data.Word_Type != '<unk>']
```

```
data
```

	Word_Type	Counts
32	,	46476
40681	the	39533
36	.	37452
33392	of	22104
40972	to	21305
...	...	...
41624	unborn	4
17765	Timbers	4
12888	Manager	4
12640	Lucy	4
11577	Jenkins	4

13750 rows × 2 columns

```
new_df = pd.concat([new_row, data]).reset_index(drop = True)
```

```
new_df
```

	Word_Type	Counts
0	<unk>	29443
1	,	46476
2	the	39533
3	.	37452
4	of	22104
...	...	...
13746	unborn	4
13747	Timbers	4
13748	Manager	4
13749	Lucy	4
13750	Jenkins	4

13751 rows × 2 columns

```
new_df = new_df.reset_index()
```

```
new_df
```

	index	Word_Type	Counts
0	0	<unk>	29443
1	1	,	46476
2	2	the	39533
3	3	.	37452
4	4	of	22104
...	...	...	...
13746	13746	unborn	4
13747	13747	Timbers	4
13748	13748	Manager	4
13749	13749	Lucy	4
13750	13750	Jenkins	4

13751 rows × 3 columns

```
print(len(new_df))
```

```
13751
```

```
new_df = new_df[['Word_Type', 'index', 'Counts']]
```

```
new_df.to_csv('vocab.txt', sep='\t', header=None, index=None)
#Check Capital letter remove in vocab and then calculate and alphanumeric ke alawa rakhna hai?
```

Task 2

```
transmission = {}
emission = {}
```

```
data_csv
```

	Index	Word_Type	POS_Tag
0	1	Pierre	NNP
1	2	Vinken	NNP
2	3	,	,
3	4	61	CD
4	5	years	NNS
...	...	...	...
912090	22	to	TO
912091	23	San	NNP
912092	24	Francisco	NNP
912093	25	instead	RB
912094	26	.	.

912095 rows × 3 columns

```
data  = data_csv.groupby(["Word_Type"])["POS_Tag"].size().reset_index(name="Counts")
```

```
data
```

	Word_Type	Counts
0	!	66
1	#	127
2	$	6762
3	%	4718
4	&	977
...	...	...
43188	zoo	1
43189	zoology	1
43190	zoomed	1

```
unk_list = list(data[data['Counts']<=3]['Word_Type'])
```

```
len(unk_list)
```

```
29443
```

```
data_csv['Word_Type'][data_csv['Word_Type'].isin(unk_list)==True] = '<unk>'
```

```
<ipython-input-31-299672d75f5a>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  data_csv['Word_Type'][data_csv['Word_Type'].isin(unk_list)==True] = '<unk>'
```

```
data_csv
```

	Index	Word_Type	POS_Tag
0	1	Pierre	NNP
1	2	<unk>	NNP
2	3	,	,
3	4	61	CD
4	5	years	NNS
...	...	...	...
912090	22	to	TO
912091	23	San	NNP
912092	24	Francisco	NNP
912093	25	instead	RB
912094	26	.	.

912095 rows × 3 columns

```
index = data_csv['Index'].values.tolist()
word_type = data_csv['Word_Type'].values.tolist()
pos_tag = data_csv['POS_Tag'].values.tolist()
```

```
count_pair = {}
count_new_senetences =  0
for i in range(len(pos_tag)-1):
    if index[i] == 1 and f'(INI, {pos_tag[i]})' in count_pair:
      count_pair[f'(INI, {pos_tag[i]})'] += 1
      count_new_senetences +=  1
    if index[i] == 1 and f'(INI, {pos_tag[i]})' not in count_pair:
      count_pair[f'(INI, {pos_tag[i]})'] = 1
    if index[i+1] == 1:
      continue
    if f'({pos_tag[i]},{pos_tag[i+1]})' in count_pair:
      count_pair[f'({pos_tag[i]},{pos_tag[i+1]})'] += 1
    else:
```

```
        count_pair[f'({pos_tag[i]},{pos_tag[i+1]})'] = 1
count_pair
```

```
{'(INI, NNP)': 7563,
 '(NNP,NNP)': 33139,
 '(NNP,,)': 12131,
 '(,,CD)': 987,
 '(CD,NNS)': 5502,
 '(NNS,JJ)': 995,
 '(JJ,,)': 1717,
 '(,,MD)': 490,
 '(MD,VB)': 7541,
 '(VB,DT)': 5661,
 '(DT,NN)': 37299,
 '(NN,IN)': 31554,
 '(IN,DT)': 31088,
 '(DT,JJ)': 17200,
 '(JJ,NN)': 26472,
 '(NN,NNP)': 1214,
 '(NNP,CD)': 1680,
 '(CD,.)': 2530,
 '(NNP,VBZ)': 3434,
 '(VBZ,NN)': 751,
 '(IN,NNP)': 14091,
 '(,,DT)': 6211,
 '(DT,NNP)': 8757,
 '(NNP,VBG)': 155,
 '(VBG,NN)': 1819,
 '(NN,.)': 13890,
 '(JJ,CC)': 1003,
 '(CC,JJ)': 2520,
 '(,,VBD)': 2396,
 '(VBD,VBN)': 2696,
 '(VBN,DT)': 1287,
 '(JJ,JJ)': 4362,
 '(INI, DT)': 8374,
 '(IN,NN)': 10362,
 '(NN,RB)': 2317,
 '(RB,VBN)': 2389,
 '(VBN,TO)': 2081,
 '(TO,VB)': 12398,
 '(VB,NNP)': 814,
 '(NNP,NN)': 5044,
 '(NN,NNS)': 10034,
 '(NNS,VBZ)': 493,
 '(VBZ,VBN)': 3093,
 '(NNS,IN)': 13569,
 '(IN,NNS)': 5672,
 '(NNS,VBN)': 1211,
 '(TO,PRP)': 107,
 '(PRP,RBR)': 25,
 '(RBR,IN)': 384,
 '(IN,CD)': 5621,
 '(IN,,)': 239,
 '(,,NNS)': 1270,
 '(NNS,VBD)': 4123,
 '(VBD,.)': 1308,
 '(NN,NN)': 15571,
 '(NN,,)': 14485,
 '(,,NN)': 2283,
 '(,,VBZ)': 1456,
```

```
pos_tag_count = {}
for i in range(len(pos_tag)):
    if pos_tag[i] in pos_tag_count:
      pos_tag_count[pos_tag[i]] += 1
    else:
      pos_tag_count[pos_tag[i]] = 1
pos_tag_count
```

```
{'NNP': 87608,
 ',': 46480,
 'CD': 34876,
 'NNS': 57859,
 'JJ': 58944,
 'MD': 9437,
 'VB': 25489,
 'DT': 78775,
 'NN': 127534,
 'IN': 94758,
 '.': 37883,
 'VBZ': 20982,
 'VBG': 14348,
 'CC': 22817,
 'VBD': 28309,
 'VBN': 19330,
 'RB': 29621,
 'TO': 21461,
 'PRP': 16766,
 'RBR': 1675,
```

```
 'WDT': 4194,
 'VBP': 12326,
 'RP': 2515,
 'PRP$': 7989,
 'JJS': 1867,
 'POS': 8284,
 '``': 6782,
 'EX': 833,
 "''": 6622,
 'WP': 2285,
 ':': 4680,
 'JJR': 3174,
 'WRB': 2050,
 '$': 6937,
 'NNPS': 2505,
 'WP$': 166,
 '-LRB-': 1305,
 '-RRB-': 1321,
 'PDT': 333,
 'RBS': 435,
 'FW': 224,
 'UH': 87,
 'SYM': 55,
 'LS': 47,
 '#': 127}
```

```
tag_to_word = {}
for i in range(len(pos_tag)):
    if f'({pos_tag[i]},{word_type[i]})' in tag_to_word:
      tag_to_word[f'({pos_tag[i]},{word_type[i]})'] += 1
    else:
      tag_to_word[f'({pos_tag[i]},{word_type[i]})'] = 1

tag_to_word
```

```
{'(NNP,Pierre)': 6,
 '(NNP,<unk>)': 10714,
 '(,,,)': 46476,
 '(CD,61)': 25,
 '(NNS,years)': 1130,
 '(JJ,old)': 213,
 '(MD,will)': 2962,
 '(VB,join)': 40,
 '(DT,the)': 39517,
 '(NN,board)': 297,
 '(IN,as)': 3354,
 '(DT,a)': 18445,
 '(JJ,nonexecutive)': 6,
 '(NN,director)': 309,
 '(NNP,Nov.)': 234,
 '(CD,29)': 74,
 '(.,.)': 37452,
 '(NNP,Mr.)': 3856,
 '(VBZ,is)': 6733,
 '(NN,chairman)': 429,
 '(IN,of)': 22100,
 '(NNP,N.V.)': 13,
 '(NNP,Dutch)': 8,
 '(VBG,publishing)': 14,
 '(NN,group)': 603,
 '(NNP,Rudolph)': 8,
 '(CD,55)': 54,
 '(CC,and)': 15338,
 '(JJ,former)': 258,
 '(NNP,Consolidated)': 14,
 '(NNP,Gold)': 13,
 '(NNP,Fields)': 3,
 '(NNP,PLC)': 105,
 '(VBD,was)': 3615,
 '(VBN,named)': 167,
 '(DT,this)': 1842,
 '(JJ,British)': 193,
 '(JJ,industrial)': 125,
 '(NN,conglomerate)': 18,
 '(DT,A)': 821,
 '(NN,form)': 80,
 '(NN,asbestos)': 26,
 '(RB,once)': 134,
 '(VBN,used)': 256,
 '(TO,to)': 21302,
 '(VB,make)': 506,
 '(NNP,Kent)': 11,
 '(NN,cigarette)': 17,
 '(NNS,filters)': 9,
 '(VBZ,has)': 3182,
 '(VBN,caused)': 53,
 '(JJ,high)': 305,
 '(NN,percentage)': 122,
```

```
        '(NN,cancer)': 71,
        '(NNS,deaths)': 29,
        '(IN,among)': 287,
        '(NNS,workers)': 197,
        '(VBN,exposed)': 16
```

```
for i in range(len(index)):
    if index[i] == 1:
      transmission[f'(INI, {pos_tag[i]})'] = count_pair[(f'(INI, {pos_tag[i]})')] / count_new_senetences
    # if i!=len(index) and index[i+1] == 1:
    #   continue
    if index[i-1]+1 == index[i]:
      transmission[f'({pos_tag[i-1]},{pos_tag[i]})'] = count_pair[f'({pos_tag[i-1]},{pos_tag[i]})'] / pos_tag_count[pos_tag[i-1]]
    # print(transmission)
    emission[f'({pos_tag[i]},{word_type[i]})'] = tag_to_word[f'({pos_tag[i]},{word_type[i]})'] / pos_tag_count[pos_tag[i]]


# for i in range(len(index)):
#     if index[i] == 1:
#       transmission[('INI', pos_tag[i])] = count_pair[('INI', pos_tag[i])] / count_new_senetences
#     # if i!=len(index) and index[i+1] == 1:
#     #   continue
#     if index[i-1]+1 == index[i]:
#       transmission[(pos_tag[i-1], pos_tag[i])] = count_pair[(pos_tag[i-1], pos_tag[i])] / pos_tag_count[pos_tag[i-1]]
#     # print(transmission)
#     emission[(pos_tag[i],word_type[i])] = tag_to_word[(pos_tag[i],word_type[i])] / pos_tag_count[pos_tag[i]]
```

```
len(transmission)
```

```
1392
```

```
len(emission)
```

```
19626
```

```
import json
with open('hmm.json', 'a', encoding="utf-8") as file:
    x = json.dumps(emission, indent=4)
    file.write(x + '\n')
with open('hmm.json', 'a', encoding="utf-8") as file:
    x = json.dumps(transmission, indent=4)
    file.write(x + '\n')
```

```
print("Number of transmission parameters: "+ str(len(transmission)))
print("Number of emission parameters: "+ str(len(emission)))
```

```
Number of transmission parameters: 1392
Number of emission parameters: 19626
```

Task 3

```
data_dev = pd.read_csv('/content/drive/MyDrive/hw2/data/dev', on_bad_lines='skip', sep='\t', low_memory=False, names=["Index", "Word_Typ
data_dev
```

	Index	Word_Type	POS_Tag
0	1	The	DT
1	2	Arizona	NNP
2	3	Corporations	NNP
3	4	Commission	NNP
4	5	authorized	VBD
...	...	...	...
131763	13	join	VB
131764	14	the	DT
131765	15	winning	VBG
131766	16	bidder	NN
131767	17	.	.

131768 rows × 3 columns

```
count_pair = {}
count_new_senetences =  0
for i in range(len(pos_tag)-1):
    if index[i] == 1 and ('INI',pos_tag[i]) in count_pair:
      count_pair[('INI', pos_tag[i])] += 1
      count_new_senetences +=  1
    if index[i] == 1 and ('INI', pos_tag[i]) not in count_pair:
      count_pair[('INI', pos_tag[i])] = 1
      count_new_senetences +=  1
    if index[i+1] == 1:
      continue
    if (pos_tag[i], pos_tag[i+1]) in count_pair:
      count_pair[(pos_tag[i],pos_tag[i+1])] += 1
    else:
      count_pair[(pos_tag[i],pos_tag[i+1])] = 1



pos_tag_count = {}
for i in range(len(pos_tag)):
    if pos_tag[i] in pos_tag_count:
      pos_tag_count[pos_tag[i]] += 1
    else:
      pos_tag_count[pos_tag[i]] = 1


tag_to_word = {}
for i in range(len(pos_tag)):
    if (pos_tag[i],word_type[i]) in tag_to_word:
      tag_to_word[(pos_tag[i],word_type[i])] += 1
    else:
      tag_to_word[(pos_tag[i],word_type[i])] = 1



for i in range(len(index)):
    if index[i] == 1:
      transmission[('INI', pos_tag[i])] = count_pair[('INI', pos_tag[i])] / count_new_senetences
    # if i!=len(index) and index[i+1] == 1:
    #   continue
    if index[i-1]+1 == index[i]:
      transmission[(pos_tag[i-1], pos_tag[i])] = count_pair[(pos_tag[i-1], pos_tag[i])] / pos_tag_count[pos_tag[i-1]]
    # print(transmission)
    emission[(pos_tag[i],word_type[i])] = tag_to_word[(pos_tag[i],word_type[i])] / pos_tag_count[pos_tag[i]]


index_dev = data_dev['Index'].values.tolist()
word_type_dev = data_dev['Word_Type'].values.tolist()
pos_tag_dev = data_dev['POS_Tag'].values.tolist()
predicted_tag = {}


def calc_emission_data(word):
  trans_check = []
  for j in emission:
    if j[1] == word:
      trans_check.append(j)
  if len(trans_check) == 0:
    for j in emission:
      if j[1] == '<unk>':
        trans_check.append(j)
  return trans_check



def calc_trans_data(prev_tag, list_emiss):
  trans_list = []
  for i in list_emiss:
    if (prev_tag,i[0]) in transmission:
      trans_list.append((prev_tag,i[0]))
  return trans_list



def max_prob_list(trans_list):
  list_of_probabilities = []
  for m in range(len(trans_list)):
    # if transmission[trans_list[m]]*emission[trans_check[m]] != []:
    list_of_probabilities.append(transmission[trans_list[m]]*emission[trans_check[m]])
    # else:
      # list_of_probabilities.append(0)
  if len(list_of_probabilities) == 0:
    return ','
  return trans_list[list_of_probabilities.index(max(list_of_probabilities))][1]
```

```
predicted_tag = {}


for i in range(len(index_dev)):
  # print(word_type_dev[i])
  if index_dev[i] == 1:
    trans_check = calc_emission_data(word_type_dev[i])
    trans_list = calc_trans_data('INI', trans_check)
    # print(trans_check)
    # print(trans_list)
    predicted_tag[i] = max_prob_list(trans_list)
  else:
    trans_check = calc_emission_data(word_type_dev[i])
    trans_list = calc_trans_data(predicted_tag[i-1], trans_check)
    # print(trans_check)
    # print(trans_list)
    predicted_tag[i] = max_prob_list(trans_list)



predicted_tag
```

```
     {0: 'DT',
      1: 'NNP',
      2: 'NNP',
      3: 'NNP',
      4: 'VBD',
      5: 'DT',
      6: 'CD',
      7: 'NN',
      8: 'NN',
      9: 'NN',
      10: 'IN',
      11: 'NNP',
      12: 'NNP',
      13: 'NNP',
      14: 'NNP',
      15: ',',
      16: 'RB',
      17: 'JJR',
      18: 'IN',
      19: 'JJ',
      20: 'JJ',
      21: 'NN',
      22: 'IN',
      23: 'DT',
      24: 'NN',
      25: 'NN',
      26: 'NN',
      27: 'CC',
      28: 'RB',
      29: 'PDT',
      30: 'DT',
      31: 'NN',
      32: 'VBD',
      33: 'IN',
      34: 'DT',
      35: 'NN',
      36: '.',
      37: 'DT',
      38: 'NN',
      39: 'VBZ',
      40: 'DT',
      41: 'NN',
      42: 'IN',
      43: 'NNS',
      44: 'IN',
      45: 'NNP',
      46: 'NNP',
      47: ',',
      48: 'VBG',
      49: 'JJ',
      50: 'NNS',
      51: ',',
      52: 'DT',
      53: 'CD',
      54: 'NN',
      55: 'VBP',
      56: 'IN',
      57: 'DT',
```

```
len(predicted_tag)
```

```
131768
```

```
correct = 0
miss = 0
for i in range(len(predicted_tag)):
  if pos_tag_dev[i] == predicted_tag[i]:
    correct+=1
print(correct/len(predicted_tag))
```

```
0.9177038431182077
```

```
predicted = []


for i in range(len(predicted_tag)):
  predicted.append(predicted_tag[i])


predicted
```

```
['DT',
 'NNP',
 'NNP',
 'NNP',
 'VBD',
 'DT',
 'CD',
 'NN',
 'NN',
 'NN',
 'IN',
 'NNP',
 'NNP',
 'NNP',
 'NNP',
 ',',
 'RB',
 'JJR',
 'IN',
 'JJ',
 'JJ',
 'NN',
 'IN',
 'DT',
 'NN',
 'NN',
 'NN',
 'CC',
 'RB',
 'PDT',
 'DT',
 'NN',
 'VBD',
 'IN',
 'DT',
 'NN',
 '.',
 'DT',
 'NN',
 'VBZ',
 'DT',
 'NN',
 'IN',
 'NNS',
 'IN',
 'NNP',
 'NNP',
 ',',
 'VBG',
 'JJ',
 'NNS',
 ',',
 'DT',
 'CD',
 'NN',
 'VBP',
 'IN',
 'DT',
```

```
greedy_out = pd.DataFrame(
    {'Index': index_dev, 'Word_Type': word_type_dev, 'Predicted_POS': predicted
    })


greedy_out
```

	Index	Word_Type	Predicted_POS
0	1	The	DT
1	2	Arizona	NNP
2	3	Corporations	NNP
3	4	Commission	NNP
4	5	authorized	VBD
...	...	...	...
131763	13	join	VB
131764	14	the	DT
131765	15	winning	VBG
131766	16	bidder	NN
131767	17		

```
greedy_out.to_csv('greedy.out', sep='\t', header=None, index=None)
```

```
greedy_out.to_csv('greedy_out.text', sep='\t', header=None, index=None)
```

```
data_test = pd.read_csv('/content/drive/MyDrive/hw2/data/test', on_bad_lines='skip', sep='\t', low_memory=False, names=["Index", "Word_T
```

```
index_test = data_test['Index'].values.tolist()
word_type_test = data_test['Word_Type'].values.tolist()
predicted_tag = {}
```

```
for i in range(len(index_test)):
  if index_test[i] == 1:
    # print(word_type_test[i])
    trans_check = calc_emission_data(word_type_test[i])
    trans_list = calc_trans_data('INI', trans_check)
    # print(trans_check)
    # print(trans_list)
    predicted_tag[i] = max_prob_list(trans_list)
  else:
    trans_check = calc_emission_data(word_type_test[i])
    trans_list = calc_trans_data(predicted_tag[i-1], trans_check)
    # print(trans_check)
    # print(trans_list)
    predicted_tag[i] = max_prob_list(trans_list)
```

```
predicted_tag
```

```
{0: 'NNP',
 1: 'NNS',
 2: 'IN',
 3: 'DT',
 4: 'NNP',
 5: 'NNPS',
 6: 'CC',
 7: 'NNP',
 8: 'NNP',
 9: 'VBD',
 10: 'NN',
 11: 'IN',
 12: 'MD',
 13: 'VB',
 14: 'WRB',
 15: 'DT',
 16: 'JJ',
 17: 'NN',
 18: 'NN',
 19: 'NN',
 20: 'MD',
 21: 'VB',
 22: 'NN',
 23: ',',
 24: 'VBG',
 25: 'DT',
 26: 'JJ',
```

```
        27: 'NN',
        28: 'TO',
        29: 'DT',
        30: 'NN',
        31: 'POS',
        32: 'NN',
        33: 'IN',
        34: 'JJ',
        35: 'NNS',
        36: '.',
        37: 'DT',
        38: 'NN',
        39: ',',
        40: 'WP$',
        41: 'NNS',
        42: 'VBP',
        43: 'NNP',
        44: 'NNP',
        45: 'NNP',
        46: '-LRB-',
        47: 'NNP',
        48: ',',
        49: 'NNP',
        50: '-RRB-',
        51: ',',
        52: 'MD',
        53: 'VB',
        54: 'DT',
        55: 'NNP',
        56: 'NNP',
```

```
predicted = []


for i in range(len(predicted_tag)):
  predicted.append(predicted_tag[i])


greedy_out = pd.DataFrame(
    {'Index': index_test, 'Word_Type': word_type_test, 'Predicted_POS': predicted
    })


greedy_out.to_csv('greedy.out', sep='\t', header=None, index=None)



greedy_out.to_csv('greedy_out.text', sep='\t', header=None, index=None)
```

Viterbi

```
actual_matched = 0
wrong_matched = 0


pos_tag_count['INI'] = count_new_senetences


pos_tag_count
```

```
     {'NNP': 87608,
      ',': 46480,
      'CD': 34876,
      'NNS': 57859,
      'JJ': 58944,
      'MD': 9437,
      'VB': 25489,
      'DT': 78775,
      'NN': 127534,
      'IN': 94758,
      '.': 37883,
      'VBZ': 20982,
      'VBG': 14348,
      'CC': 22817,
      'VBD': 28309,
      'VBN': 19330,
      'RB': 29621,
      'TO': 21461,
      'PRP': 16766,
      'RBR': 1675,
      'WDT': 4194,
```

```
        'VBP': 12326,
        'RP': 2515,
        'PRP$': 7989,
        'JJS': 1867,
        'POS': 8284,
        '``': 6782,
        'EX': 833,
        "''": 6622,
        'WP': 2285,
        ':': 4680,
        'JJR': 3174,
        'WRB': 2050,
        '$': 6937,
        'NNPS': 2505,
        'WP$': 166,
        '-LRB-': 1305,
        '-RRB-': 1321,
        'PDT': 333,
        'RBS': 435,
        'FW': 224,
        'UH': 87,
        'SYM': 55,
        'LS': 47,
        '#': 127,
        'INI': 38218}


pos_tag_index = dict()
index = 0


start_index = -1
end_index = -1


for key in pos_tag_count:
    if key == ".":
        end_index = index
    if key == 'INI':
        start_index = index
    pos_tag_index[index] = key
    index += 1


dev = open('/content/drive/MyDrive/hw2/data/dev','r')


words_list = list()
actual_tag = list()


data_lines = dev.readlines()


def calc_max(predict_tag_array):
  for k in range(len(predict_tag_array)):
    if predict_tag_array[k] == actual_tag[k]:
      actual_matched += 1
    else:
      wrong_matched += 1


predd = []
for each_line in data_lines:
  words = each_line[:-1].split("\t")
  #Unless this sentence goes till the end
  if words[0]!="":
    if words[1] in unk_list:
      words_list.append('<unk>')
    else:
      words_list.append(words[1])
    actual_tag.append(words[2])

  #If we encounter a new line
  else:
    dp = [[-1 for _ in range(len(words_list))] for _ in range(len(pos_tag_count))]
    # print(dp)
    for i in range(len(pos_tag_count)):
      #the tag right now
      current_tag = pos_tag_index[i]
      transmission_attribute = ('INI', current_tag)
      transmission_probability = 0.0000000000001
      #calculating transmission
      if transmission_attribute in transmission:
        transmission_probability = transmission[transmission_attribute]
        # print(transmission_probability)
```

```
    emission_attribute = (current_tag , words_list[0])
    emission_probabiity = 0.0000000000001
    #calculating emission
    if emission_attribute in emission:
      emission_probabiity = emission[emission_attribute]
      # print(emission_probabiity)

    dp[i][0] = transmission_probability*emission_probabiity
    # print(dp)
  for word_index in range(1,len(words_list)):
    cur_word = words_list[word_index]

    for i in range(len(pos_tag_count)):
      current_tag = pos_tag_index[i]
      max_prob = 0
      emission_attribute = (current_tag , cur_word)
      emission_probabiity = 0.0000000000001
      if emission_attribute in emission:
        emission_probabiity = emission[emission_attribute]
        # print(emission_probabiity)

      for j in range(len(pos_tag_count)):
        prev_tag = pos_tag_index[j]

        transmission_attribute = (prev_tag, current_tag)
        transmission_probability = 0.0000000000001
        if transmission_attribute in transmission:
          transmission_probability = transmission[transmission_attribute]
          # print(transmission_probability)

        max_prob = max(max_prob, dp[j][word_index-1]*emission_probabiity*transmission_probability)

      dp[i][word_index] = max_prob
      # print(dp)


  predict_tag_array = []

  column = len(words_list) - 1
  next_tag = -1
  max_prob = 0
  for i in range(len(pos_tag_count)):
    if max_prob < dp[i][column]:
      max_prob = dp[i][column]
      next_tag = i

  if next_tag == -1:
    next_tag = end_index

  predict_tag_array.append(pos_tag_index[next_tag])

  for column in range(len(words_list)-2,-1,-1):
    next_word = words_list[column+1]
    max_prev_tag = start_index
    store_max_prob = max_prob
    diff = 1
    for i in range(len(pos_tag_count)):
      prev_tag = i
      cur_prob = dp[i][column]

      if cur_prob != 0:
        transmission_attribute = (pos_tag_index[prev_tag] , pos_tag_index[next_tag])
        transmission_probability = 0.0000000000001
        if transmission_attribute in transmission:
          transmission_probability = transmission[transmission_attribute]


        emission_attribute = (pos_tag_index[next_tag] , next_word)
        emission_probabiity = 0.0000000000001
        if emission_attribute in emission:
          emission_probabiity = emission[emission_attribute]

        if diff > abs(cur_prob*(transmission_probability*emission_probabiity) - max_prob):
          diff = abs(cur_prob*(transmission_probability*emission_probabiity) - max_prob)
          max_prev_tag = prev_tag
          store_max_prob = cur_prob

    next_tag = max_prev_tag
    max_prob = store_max_prob

    predict_tag_array.append(pos_tag_index[max_prev_tag])
```

```
    predict_tag_array.reverse()

    for k in range(len(predict_tag_array)):
      if predict_tag_array[k] == actual_tag[k]:
        actual_matched += 1
      else:
        wrong_matched += 1
    # predd.append(predict_tag_array)
    # words_list = []
    # actual_tag = []
    # print(predd)
```

```
['DT', 'MD', 'VB', 'PRP', 'DT', 'NN', 'TO', 'VB', 'DT', 'NN', 'CC', 'RB', 'VB', 'DT', 'JJ', 'NN', '.', 'DT', 'NNP', 'NNP', 'NNP',
['That', 'could', 'cost', 'him', 'the', 'chance', 'to', 'influence', 'the', 'outcome', 'and', 'perhaps', 'join', 'the', 'winning',
```

```
print("Viterbi - Accuracy: "+ str(actual_matched/(actual_matched+wrong_matched)))
```

```
Viterbi - Accuracy: 0.9346125595377823
```

```
actual_matched + wrong_matched
```

```
132059
```

```
actual_matched = 0
wrong_matched = 0
```

```
pos_tag_count['INI'] = count_new_senetences
```

```
pos_tag_count
```

```
{'NNP': 87608,
 ',': 46480,
 'CD': 34876,
 'NNS': 57859,
 'JJ': 58944,
 'MD': 9437,
 'VB': 25489,
 'DT': 78775,
 'NN': 127534,
 'IN': 94758,
 '.': 37883,
 'VBZ': 20982,
 'VBG': 14348,
 'CC': 22817,
 'VBD': 28309,
 'VBN': 19330,
 'RB': 29621,
 'TO': 21461,
 'PRP': 16766,
 'RBR': 1675,
 'WDT': 4194,
 'VBP': 12326,
 'RP': 2515,
 'PRP$': 7989,
 'JJS': 1867,
 'POS': 8284,
 '``': 6782,
 'EX': 833,
 "''": 6622,
 'WP': 2285,
 ':': 4680,
 'JJR': 3174,
 'WRB': 2050,
 '$': 6937,
 'NNPS': 2505,
 'WP$': 166,
 '-LRB-': 1305,
 '-RRB-': 1321,
 'PDT': 333,
 'RBS': 435,
 'FW': 224,
 'UH': 87,
 'SYM': 55,
 'LS': 47,
 '#': 127,
 'INI': 38218}
```

```
pos_tag_index = dict()
index = 0


start_index = -1
end_index = -1


for key in pos_tag_count:
    if key == ".":
        end_index = index
    if key == 'INI':
        start_index = index
    pos_tag_index[index] = key
    index += 1


test = open('/content/drive/MyDrive/hw2/data/test','r')


words_list = list()
actual_tag = list()


data_lines = test.readlines()


def calc_max(predict_tag_array):
  for k in range(len(predict_tag_array)):
    if predict_tag_array[k] == actual_tag[k]:
      actual_matched += 1
    else:
      wrong_matched += 1


output = open('viterbi.txt','w')


predd = []
for each_line in data_lines:
  words = each_line[:-1].split("\t")
  #Unless this sentence goes till the end
  if words[0]!="":
    if words[1] in unk_list:
      words_list.append('<unk>')
    else:
      words_list.append(words[1])

  #If we encounter a new line
  else:
    dp = [[-1 for _ in range(len(words_list))] for _ in range(len(pos_tag_count))]
    # print(dp)
    for i in range(len(pos_tag_count)):
      #the tag right now
      current_tag = pos_tag_index[i]
      transmission_attribute = ('INI', current_tag)
      transmission_probability = 0.0000000000001
      #calculating transmission
      if transmission_attribute in transmission:
        transmission_probability = transmission[transmission_attribute]
        # print(transmission_probability)

      emission_attribute = (current_tag , words_list[0])
      emission_probabiity = 0.0000000000001
      #calculating emission
      if emission_attribute in emission:
        emission_probabiity = emission[emission_attribute]
        # print(emission_probabiity)

      dp[i][0] = transmission_probability*emission_probabiity
      # print(dp)
    for word_index in range(1,len(words_list)):
      cur_word = words_list[word_index]

      for i in range(len(pos_tag_count)):
        current_tag = pos_tag_index[i]
        max_prob = 0
        emission_attribute = (current_tag , cur_word)
        emission_probabiity = 0.0000000000001
        if emission_attribute in emission:
          emission_probabiity = emission[emission_attribute]
          # print(emission_probabiity)

        for j in range(len(pos_tag_count)):
          prev_tag = pos_tag_index[j]
```

```
      transmission_attribute = (prev_tag, current_tag)
      transmission_probability = 0.0000000000001
      if transmission_attribute in transmission:
        transmission_probability = transmission[transmission_attribute]
        # print(transmission_probability)

      max_prob = max(max_prob, dp[j][word_index-1]*emission_probabiity*transmission_probability)

    dp[i][word_index] = max_prob
    # print(dp)


predict_tag_array = []

column = len(words_list) - 1
next_tag = -1
max_prob = 0
for i in range(len(pos_tag_count)):
  if max_prob < dp[i][column]:
    max_prob = dp[i][column]
    next_tag = i

if next_tag == -1:
  next_tag = end_index

predict_tag_array.append(pos_tag_index[next_tag])

for column in range(len(words_list)-2,-1,-1):
  next_word = words_list[column+1]
  max_prev_tag = start_index
  store_max_prob = max_prob
  diff = 1
  for i in range(len(pos_tag_count)):
    prev_tag = i
    cur_prob = dp[i][column]

    if cur_prob != 0:
      transmission_attribute = (pos_tag_index[prev_tag] , pos_tag_index[next_tag])
      transmission_probability = 0.0000000000001
      if transmission_attribute in transmission:
        transmission_probability = transmission[transmission_attribute]


      emission_attribute = (pos_tag_index[next_tag] , next_word)
      emission_probabiity = 0.0000000000001
      if emission_attribute in emission:
        emission_probabiity = emission[emission_attribute]

      if diff > abs(cur_prob*(transmission_probability*emission_probabiity) - max_prob):
        diff = abs(cur_prob*(transmission_probability*emission_probabiity) - max_prob)
        max_prev_tag = prev_tag
        store_max_prob = cur_prob

  next_tag = max_prev_tag
  max_prob = store_max_prob

  predict_tag_array.append(pos_tag_index[max_prev_tag])

predict_tag_array.reverse()

for k in range(len(predict_tag_array)):
  output.write(str(k+1)+"\t"+str(words_list[k])+"\t"+str(predict_tag_array[k])+"\n")




# for k in range(len(predict_tag_array)):
#   if predict_tag_array[k] == actual_tag[k]:
#     actual_matched += 1
#   else:
#     wrong_matched += 1
# predd.append(predict_tag_array)
words_list = []
# actual_tag = []
# print(predd)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-288-c5939cf6151e> in <module>
     47           prev_tag = pos_tag_index[j]
     48
---> 49           transmission_attribute = (prev_tag, current_tag)
     50           transmission_probability = 0.0000000000001
     51           if transmission_attribute in transmission:

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

34s completed at 8:50 AM