In [ ]:
```
import numpy as np
import os
import conlleval
import copy

train_dir = "dataset/train"
test_dir = "dataset/dev.in"

START_STATE_KEY = "START"
STOP_STATE_KEY = "STOP"

LARGE_NEG = -2**52
```

In [ ]:
```
def tokenize(file_path):
    data, lst = [], []
    with open(file_path, 'r') as f:
        for line in f:
            if line== '\n':
                data.append(lst)
                lst = []
            else:
                lines = line.replace("\n",'').split(" ")
                lst.append(tuple(lines))
    return data

train_sentences = tokenize(train_dir)
print(train_sentences[:5])
```

```
[[('All', 'O'), ('in', 'O'), ('all', 'O'), (',', 'O'), ('the', 'O'), ('food', 'B-positive'),
('was', 'O'), ('great', 'O'), ('(', 'O'), ('except', 'O'), ('for', 'O'), ('the', 'O'), ('dess
serts', 'B-negative'), (')', 'O'), ('.', 'O')], [('I', 'O'), ('have', 'O'), ('NEVER', 'O'),
('been', 'O'), ('disappointed', 'O'), ('in', 'O'), ('the', 'O'), ('Red', 'B-positive'), ('Ey
e', 'I-positive'), ('.', 'O')], [('Great', 'O'), ('food', 'B-positive'), ('with', 'O'), ('a
n', 'O'), ('awesome', 'O'), ('atmosphere', 'B-positive'), ('!', 'O')], [('The', 'O'), ('sangr
ia', 'B-positive'), ('was', 'O'), ('pretty', 'O'), ('tasty', 'O'), ('and', 'O'), ('good',
'O'), ('on', 'O'), ('a', 'O'), ('hot', 'O'), ('muggy', 'O'), ('day', 'O'), ('.', 'O')], [('Al
so', 'O'), (',', 'O'), ('waiters', 'B-negative'), ('try', 'O'), ('to', 'O'), ('push', 'O'),
('more', 'O'), ('food', 'O'), ('on', 'O'), ('you', 'O'), (',', 'O'), ('like', 'O'), ('sugges
t', 'O'), ('things', 'O'), ('as', 'O'), ('if', 'O'), ('they', 'O'), ('are', 'O'), ('complimen
tary', 'O'), ('when', 'O'), ('they', 'O'), ('actually', 'O'), ('cost', 'O'), ('$', 'O'),
('.', 'O')]]
```

PART 1

1i)

In [ ]:
```
def MLE_emission_parameters(train_sentences):
    ''' Calculates the emission parameters by count(y->x)/count(y)

    :param train_sentences: our train file tokenised sentences
    :type train_sentences: list(tuple())

    :return count_y_dict: Count of labels
    :rtype: dict()

    :return count_y_to_x_dict: Count of words and labels
    :rtype: dict()

    :param emission_dict: value of Count(labels->words)/Count(labels), keys are tuples of wor
d and label ('emission: O+All', -9.01768561), value MLE
    :rtype: dict

    '''

    count_y_dict = {}
    count_y_to_x_dict = {}
    emission_dict = {}
    word_bank = []

    for sentence in train_sentences:
        for x_y_pair in sentence:
            word, label = x_y_pair
            word_bank.append(word)
            count_y_dict[label] = count_y_dict.get(label,0) + 1
            count_y_to_x_dict[(label,word)] = count_y_to_x_dict.get((label,word),0) + 1

    # Calculate our emission
    for key, value in count_y_to_x_dict.items():
        label = key[0]
        word = key[1]
        string = f"emission: {label}+{word}"
        prob =  value / count_y_dict.get(label)
        emission_dict[string] = float(np.where(prob != 0, np.log(prob), 0))

    # handle missing possible words and labels
    unique_word_bank, labels = set(word_bank), count_y_dict.keys()
    for label in labels:
        for word in unique_word_bank:
            string = f"emission: {label}+{word}"
            if string not in emission_dict:
                emission_dict[string] = LARGE_NEG

    return count_y_dict, count_y_to_x_dict, emission_dict
```

In [ ]:
```
count_y_dict, count_y_to_x_dict, emission_dict = MLE_emission_parameters(train_sentences)
print(list(emission_dict.items())[0:5])
```

```
[('emission: O+All', -9.017685611042436), ('emission: O+in', -4.54034879656423), ('emission:
O+all', -5.785564559424215), ('emission: O+,', -3.24728344904484), ('emission: O+the', -3.091
6488692569097)]
```

1ii)

In [ ]:

```
def  MLE_transition_parameters(train_dir, emission_dict):
    ''' Calculates the transition parameters by count(y->y-1)/count(y)

    :param train_dir: our train file
    :type train_sentences: str

    :param emission_dict: Count(y->x)/Count(y), keys are tuples of word and label ('emission:
O+All', -9.01768561), value MLE
    :type emission_dict: dict()

    :return count_y_to_y_dict: Count of labels and previous label
    :rtype: dict()

    :return emission_transition_dict: value of Count(labels->words)/Count(labels) for emissio
n and Count(prev_labels->labels)/Count(labels) for transmission, keys are tuples of word and
 label ('emission: O+All', -9.01768561), value MLE
    :rtype: dict()
    '''
    count_y_dict = {}
    count_y_to_y_dict = {}
    prev_label = ""
    emission_transition_dict = copy.deepcopy(emission_dict)

    with open(train_dir, "r", encoding="utf8") as f:
        for line in f:
            # Parse each line
            if len(line.split(" ")) == 2:
                _, label = line.replace("\n","").split(" ")
            else:
                label = ''
            if label == '' and prev_label != '':
                count_y_dict[STOP_STATE_KEY] = count_y_dict.get(STOP_STATE_KEY) + 1 if count_
y_dict.get(STOP_STATE_KEY) else 1
            elif label !='':
                if prev_label == '':
                    count_y_dict[START_STATE_KEY] = count_y_dict.get(START_STATE_KEY) + 1 if
count_y_dict.get(START_STATE_KEY) else 1
                if label in count_y_dict:
                    count_y_dict[label] = count_y_dict.get(label)+1
                else:
                    count_y_dict[label] = 1
            if prev_label == '' and label != '':
                if (START_STATE_KEY, label) in count_y_to_y_dict:
                    count_y_to_y_dict[(START_STATE_KEY, label)] = count_y_to_y_dict.get((STAR
T_STATE_KEY, label)) + 1
                else:
                    count_y_to_y_dict[(START_STATE_KEY, label)] = 1
            elif label == '' and prev_label != '':
                if (prev_label, STOP_STATE_KEY) in count_y_to_y_dict:
                    count_y_to_y_dict[(prev_label, STOP_STATE_KEY)] = count_y_to_y_dict.get((
prev_label, STOP_STATE_KEY)) + 1
                else:
                    count_y_to_y_dict[(prev_label, STOP_STATE_KEY)] = 1
            elif label != '' and prev_label != '':
                if (prev_label, label) in count_y_to_y_dict:
                    count_y_to_y_dict[(prev_label, label)] = count_y_to_y_dict.get((prev_labe
l, label)) + 1
                else:
                    count_y_to_y_dict[(prev_label, label)] = 1
            prev_label = label

    # Calculate our transition
    for key, value in count_y_to_y_dict.items(): # Default is iterate keys()
        prev_label = key[0]
```

```
        label = key[1]
        string = f"transition: {prev_label}+{label}"
        prob =  value / count_y_dict.get(prev_label)
        emission_transition_dict[string] = float(np.where(prob != 0, np.log(prob), 0))
    # print("MLE: \n",list(emission_transition_dict.items()), len(emission_transition_dict)
,"\n")

    labels = count_y_dict.keys()
    for label in labels:
        for prev_label in labels:
            string = f"transition: {prev_label}+{label}"
            if string not in emission_transition_dict:
                emission_transition_dict[string] = LARGE_NEG

    return count_y_to_y_dict, emission_transition_dict
```

In [ ]:
```
count_y_to_y_dict, emission_transition_dict = MLE_transition_parameters(train_dir, emission_d
ict)
print(list(emission_transition_dict.items())[:5])
print(list(emission_transition_dict.items())[-5:])
```

```
[('emission: O+All', -9.017685611042436), ('emission: O+in', -4.54034879656423), ('emission:
O+all', -5.785564559424215), ('emission: O+,', -3.24728344904484), ('emission: O+the', -3.091
6488692569097)]
[('transition: B-positive+I-negative', -4503599627370496), ('transition: STOP+I-negative', -4
503599627370496), ('transition: I-positive+I-negative', -4503599627370496), ('transition: B-n
eutral+I-negative', -4503599627370496), ('transition: I-neutral+I-negative', -450359962737049
6)]
```

Part 2

2i)

In [ ]:
```
def score(sentence, emission_transition_dict):
    ''' Calculates the score with of a given pair based on emission and transmission features

    :param sentences: our  file tokenised sentences
    :type sentences: list(tuple())

    :return emission_transition_dict: value of Count(labels->words)/Count(labels) for emissio
n and Count(prev_labels->labels)/Count(labels) for transmission, keys are tuples of word and
 label ('emission: O+All', -9.01768561), value MLE
    :type emission_transition_dict: dict()

    :param score: our emission score + transition score for sentence
    :type sentences: float
    '''
    score = 0
    emission_score = 0
    transition_score = 0
    x_seq = [x[0] for x in sentence]
    y_seq = [START_STATE_KEY]+[y[1] for y in sentence]+[STOP_STATE_KEY]

    for i in range(len(x_seq)):
        label = y_seq[i+1]
        word = x_seq[i]
        key = f"emission: {label}+{word}"
        emission_score += emission_transition_dict[key]
    for j in range(1, len(y_seq)):
        prev_label = y_seq[j-1]
        label = y_seq[j]
        key = f"transition: {prev_label}+{label}"
        transition_score += emission_transition_dict[key]
    score = emission_score + transition_score
    return score
```

In [ ]:
```
score(train_sentences[0],emission_transition_dict)
```

Out[ ]:
```
-85.52845366888094
```

2ii)

In [ ]:
```
test_sentences = tokenize(test_dir)
print(test_sentences[:5])
```

```
[[('Loved',), ('it',)], [('The',), ('music',), ('playing',), ('was',), ('very',), ('hip',),
(',',), ('20-30',), ('something',), ('pop',), ('music',), (',',), ('but',), ('the',), ('subwo
ofer',), ('to',), ('the',), ('sound',), ('system',), ('was',), ('located',), ('under',), ('m
y',), ('seat',), (',',), ('which',), ('became',), ('annoying',), ('midway',), ('through',),
('dinner',), ('.',)], [('This',), ('place',), ('has',), ('ruined',), ('me',), ('for',), ('nei
ghborhood',), ('sushi',), ('.',)], [('I',), ('have',), ('never',), ('eaten',), ('in',), ('th
e',), ('restaurant',), (',',), ('however',), (',',), ('upon',), ('reading',), ('the',), ('rev
iews',), ('I',), ('got',), ('take',), ('out',), ('last',), ('week',), ('.',)], [('It',), ("is
n't",), ('the',), ('cheapest',), ('sushi',), ('but',), ('has',), ('been',), ('worth',), ('i
t',), ('every',), ('time',), ('.',)]]
```

In [ ]:
```
def viterbi_algo(test_sentences, count_y_dict, emission_transition_dict, output_name):
    ''' Decoding process that finds globally finds the best possible labels from past MLE sco
res, saves file to output folder

    :param test_sentences: our file tokenised sentences
    :type test_sentences: list(tuple())

    :param count_y_dict: Count of labels
    :param count_y_dict: dict()

    :param emission_transition_dict: value of Count(labels->words)/Count(labels) for emission
and Count(prev_labels->labels)/Count(labels) for transmission, keys are tuples of word and la
bel ('emission: O+All', -9.01768561), value MLE
    :param emission_transition_dict: dict()
    '''

    pi = [{}]
    path = {}
    labels = count_y_dict.keys()
    os.makedirs('output',exist_ok=True)

    with open(f'output/{output_name}', "w") as outfile:
        for sentence in test_sentences:
            # j = 0 (START)
            for label in labels:
                pi[0][label] = emission_transition_dict.get(f"transition: {START_STATE_KEY}+{
label}",LARGE_NEG) + emission_transition_dict.get(f"emission: {label}+{sentence[0][0]}",LARGE
_NEG)
                path[label] = [label]
            # j = 1 to N-1
            for idx in range(1,len(sentence)):
                pi.append({})
                newpath = {}
                for label_y in labels:
                    (prob, label) = max([(pi[idx-1][prev_label] + emission_transition_dict.ge
t(f"transition: {prev_label}+{label_y}",LARGE_NEG) + emission_transition_dict.get(f"emission:
{label_y}+{sentence[idx][0]}",LARGE_NEG), prev_label)
                                         for prev_label in labels])
                    pi[idx][label_y] = prob
                    newpath[label_y] = path[label] + [label_y]
                path = newpath
            # j = N (STOP)
            idx = len(sentence)
            (prob, label) = max([(pi[idx-1][label_y] + emission_transition_dict.get(f"transit
ion: {label_y}+{STOP_STATE_KEY}", LARGE_NEG), label_y) for label_y in labels])

            # handle inconsistent length
            if len(sentence) != len(path[label]):
                print(len(sentence),len(path[label]))
                raise Exception("{} has a different lenght with {}".format(sentence, path[lab
el]))

            # write to file
            for i in range(len(sentence)):
                line = "{} {}\n".format(sentence[i][0], path[label][i])
                outfile.write(line)

            outfile.write("\n")
```

In [ ]:
```
viterbi_algo(test_sentences, count_y_dict, emission_transition_dict, output_name='dev.p2.out'
)
```

Evaluation of dev.p2.out

In [ ]:
```
prediction_dir = 'output/dev.p2.out'
truth_dir = 'dataset/dev.out'

def evaluate_results(truth_dir,prediction_dir):
    predictions = []
    prediction_sentences = tokenize(prediction_dir)
    for sentence in prediction_sentences:
        for word_pair in sentence:
            predictions.append(word_pair[1])
    lines = """"""
    idx = 0
    with open(truth_dir, "r", encoding="utf8") as tstr:
        for line in tstr:
            if len(line) > 1:
                newline = line.replace("\n",f" {predictions[idx]}\n")
                lines += newline
                idx += 1
            else:
                lines += "\n"
    return lines.splitlines()

lines = evaluate_results(truth_dir,prediction_dir)
res = conlleval.evaluate(lines)
print(conlleval.report(res))
```

```
processed 3809 tokens with 210 phrases; found: 132 phrases; correct: 63.
accuracy:  93.23%; precision:  47.73%; recall:  30.00%; FB1:  36.84
         negative: precision:  35.29%; recall:   9.23%; FB1:  14.63  17
          neutral: precision:  20.00%; recall:  12.50%; FB1:  15.38  5
         positive: precision:  50.91%; recall:  40.88%; FB1:  45.34  110
```

Part 3

3i)

In [ ]:

```
def logSumExp(a):
    max = np.max(a)
    sumOfExp = np.exp(a - max).sum()
    return max + np.log(sumOfExp)

def forward_algorithm(sentence, count_y_dict, emission_transition_dict):
    pi = [{}]
    labels = count_y_dict.keys()
    # j = 0 (START)
    for label in labels:
        pi[0][label] = emission_transition_dict.get(f"transition: {START_STATE_KEY}+{label}",
LARGE_NEG) + emission_transition_dict.get(f"emission: {label}+{sentence[0][0]}",LARGE_NEG)

    # j = 1 to N-1
    for idx in range(1,len(sentence)):
        pi.append({})

        for label in labels:
            log_a = []
            for prev_label in labels:
                log_a.append(pi[idx-1][prev_label] + emission_transition_dict.get(f"transitio
n: {prev_label}+{label}",LARGE_NEG) + emission_transition_dict.get(f"emission: {label}+{sente
nce[idx][0]}",LARGE_NEG))
            pi[idx][label] = logSumExp(log_a)

    # j = N (STOP)
    idx = len(sentence)
    log_a = []
    for label in labels:
        log_a.append(pi[idx-1][label] + emission_transition_dict.get(f"transition: {label}+{S
TOP_STATE_KEY}", LARGE_NEG))
    return pi, logSumExp(log_a)

forward_algorithm(train_sentences[0], count_y_dict, emission_transition_dict)
```

Out[ ]:
```
([{'O': -9.079345204990318,
   'B-positive': -4503599627370499.0,
   'B-negative': -4503599627370501.0,
   'I-positive': -9007199254740992,
   'B-neutral': -4503599627370501.0,
   'I-neutral': -9007199254740992,
   'I-negative': -9007199254740992},
  {'O': -13.766622979615455,
   'B-positive': -4503599627370508.0,
   'B-negative': -4503599627370509.0,
   'I-positive': -4503599627370506.0,
   'B-neutral': -4503599627370511.0,
   'I-neutral': -9007199254741000.0,
   'I-negative': -4503599627370507.0},
  {'O': -19.699116517100578,
   'B-positive': -23.953109923204384,
   'B-negative': -23.93728208854489,
   'I-positive': -9007199254741004.0,
   'B-neutral': -4503599627370516.0,
   'I-neutral': -9007199254741006.0,
   'I-negative': -9007199254741004.0},
  {'O': -23.068893163028275,
   'B-positive': -4503599627370519.0,
   'B-negative': -4503599627370520.0,
   'I-positive': -30.46462428231426,
   'B-neutral': -4503599627370522.0,
   'I-neutral': -9007199254741012.0,
   'I-negative': -4503599627370522.0},
  {'O': -26.307057565319838,
   'B-positive': -4503599627370522.0,
   'B-negative': -4503599627370523.0,
   'I-positive': -36.10781617712779,
   'B-neutral': -4503599627370525.0,
   'I-neutral': -9007199254741016.0,
   'I-negative': -4503599627370523.0},
  {'O': -32.599955203725884,
   'B-positive': -31.618347185707613,
   'B-negative': -32.86679876160505,
   'I-positive': -40.18239215402747,
   'B-neutral': -34.10786696720044,
   'I-neutral': -4503599627370525.0,
   'I-negative': -4503599627370527.0},
  {'O': -35.36430708731581,
   'B-positive': -4503599627370532.0,
   'B-negative': -4503599627370533.0,
   'I-positive': -4503599627370529.0,
   'B-neutral': -4503599627370535.0,
   'I-neutral': -4503599627370532.0,
   'I-negative': -4503599627370530.0},
  {'O': -40.60365344424098,
   'B-positive': -4503599627370534.0,
   'B-negative': -4503599627370536.0,
   'I-positive': -9007199254741026.0,
   'B-neutral': -4503599627370537.0,
   'I-neutral': -9007199254741028.0,
   'I-negative': -9007199254741028.0},
  {'O': -46.510171495322844,
   'B-positive': -4503599627370540.0,
   'B-negative': -4503599627370541.0,
   'I-positive': -4503599627370539.0,
   'B-neutral': -4503599627370543.0,
   'I-neutral': -9007199254741032.0,
   'I-negative': -9007199254741032.0},
  {'O': -55.38710401197441,
```

```
 'B-positive': -4503599627370546.0,
 'B-negative': -4503599627370547.0,
 'I-positive': -9007199254741036.0,
 'B-neutral': -4503599627370548.0,
 'I-neutral': -9007199254741040.0,
 'I-negative': -9007199254741040.0},
{'O': -60.00135665158466,
 'B-positive': -4503599627370554.0,
 'B-negative': -4503599627370556.0,
 'I-positive': -4503599627370553.0,
 'B-neutral': -4503599627370557.0,
 'I-neutral': -9007199254741046.0,
 'I-negative': -4503599627370554.0},
{'O': -63.23993449890247,
 'B-positive': -4503599627370559.0,
 'B-negative': -4503599627370560.0,
 'I-positive': -4503599627370559.0,
 'B-neutral': -4503599627370562.0,
 'I-neutral': -9007199254741052.0,
 'I-negative': -4503599627370559.0},
{'O': -4503599627370559.0,
 'B-positive': -4503599627370562.0,
 'B-negative': -73.41059360783191,
 'I-positive': -9007199254741054.0,
 'B-neutral': -4503599627370565.0,
 'I-neutral': -9007199254741056.0,
 'I-negative': -9007199254741056.0},
{'O': -79.33052533092805,
 'B-positive': -9007199254741058.0,
 'B-negative': -9007199254741060.0,
 'I-positive': -4503599627370567.0,
 'B-neutral': -9007199254741060.0,
 'I-neutral': -9007199254741064.0,
 'I-negative': -4503599627370571.0},
{'O': -82.2752126601316,
 'B-positive': -4503599627370578.0,
 'B-negative': -4503599627370580.0,
 'I-positive': -9007199254741064.0,
 'B-neutral': -4503599627370581.0,
 'I-neutral': -9007199254741072.0,
 'I-negative': -9007199254741068.0}],
-84.8718181557256)
```

In [ ]:
```python
def loss_fn(sentences, count_y_dict, emission_transition_dict):
    loss = 0
    for sent in sentences:
        loss+= score(sent, emission_transition_dict)
        _, update = forward_algorithm(sent, count_y_dict, emission_transition_dict)
        loss-= update
    return (-1)*loss
loss_fn(train_sentences, count_y_dict, emission_transition_dict)
```

Out[ ]:
```
2050.74053383538
```

3ii)

In [ ]:

```
def backward_algorithm(sentence, count_y_dict, emission_transition_dict):
    pi = [{} for i in range(len(sentence))]
    labels = count_y_dict.keys()

    # j = N (STOP)
    for label in labels:
        pi[len(sentence)-1][label] = emission_transition_dict.get(f"transition: {label}+{STOP
_STATE_KEY}", LARGE_NEG)

    # j = N-1 to 1
    for idx in range(len(sentence)-1,0,-1):
        for label in labels:
            log_b = []
            for next_label in labels:
                log_b.append(pi[idx][next_label] + emission_transition_dict.get(f"transition:
{label}+{next_label}",LARGE_NEG) + emission_transition_dict.get(f"emission: {next_label}+{sen
tence[idx][0]}",LARGE_NEG))
            pi[idx-1][label] = logSumExp(log_b)

    # j = 0 (START)
    log_b = []
    for label in labels:
        log_b.append(pi[0][label] + emission_transition_dict.get(f"transition: {START_STATE_K
EY}+{label}",LARGE_NEG) + emission_transition_dict.get(f"emission: {label}+{sentence[0][0]}",
LARGE_NEG))

    return pi, logSumExp(log_b)
```

In [ ]:
```
backward_algorithm(train_sentences[0], count_y_dict, emission_transition_dict)
```

Out[ ]:
```
([{'O': -75.79247295073526,
   'B-positive': -75.96013210037187,
   'B-negative': -75.65394924920902,
   'I-positive': -76.11520236719656,
   'B-neutral': -75.87847553035473,
   'I-neutral': -76.29947044008102,
   'I-negative': -75.64641026599291},
  {'O': -71.10519517611013,
   'B-positive': -71.34020585812414,
   'B-negative': -71.21765957710079,
   'I-positive': -71.52518608374271,
   'B-neutral': -71.21614051255226,
   'I-neutral': -71.63713542227855,
   'I-negative': -71.49666270104414},
  {'O': -65.19764439544767,
   'B-positive': -65.36383891214948,
   'B-negative': -65.28516603961566,
   'I-positive': -65.51792969991837,
   'B-neutral': -65.28364697506713,
   'I-neutral': -65.70464188479342,
   'I-negative': -65.56416916355901},
  {'O': -61.80343196834192,
   'B-positive': -61.890165966505435,
   'B-negative': -61.831706821012546,
   'I-positive': -61.994494944318916,
   'B-neutral': -61.889434547961386,
   'I-neutral': -62.310429457687675,
   'I-negative': -62.020553032988104},
  {'O': -58.564854121024105,
   'B-positive': -58.32274068137175,
   'B-negative': -59.55602438373126,
   'I-positive': -58.04127044988653,
   'B-neutral': -58.830965146599674,
   'I-neutral': -58.0427976934884,
   'I-negative': -59.33749196269587},
  {'O': -53.674913383749015,
   'B-positive': -53.88525725696903,
   'B-negative': -53.76243502791701,
   'I-positive': -54.06996153455894,
   'B-neutral': -53.76091596336849,
   'I-neutral': -54.18191087309478,
   'I-negative': -54.041438151860376},
  {'O': -49.50751106840979,
   'B-positive': -49.717854941629795,
   'B-negative': -49.59503271257778,
   'I-positive': -49.90255921921971,
   'B-neutral': -49.59351364802926,
   'I-neutral': -50.01450855775555,
   'I-negative': -49.874035836521145},
  {'O': -44.268164711484616,
   'B-positive': -43.61902437617982,
   'B-negative': -44.35568635565261,
   'I-positive': -43.45656805803591,
   'B-neutral': -44.35416729110408,
   'I-neutral': -44.77516220083037,
   'I-negative': -44.63468947959597},
  {'O': -38.361646660402755,
   'B-positive': -38.57199053362276,
   'B-negative': -38.44916830457075,
   'I-positive': -38.756694811212675,
   'B-neutral': -38.44764924002222,
   'I-neutral': -38.86864414974851,
   'I-negative': -38.72817142851411},
  {'O': -29.48471414375119,
```

```
  'B-positive': -29.61520927208044,
  'B-negative': -29.458136952015625,
  'I-positive': -29.746145856385503,
  'B-neutral': -29.57071672337066,
  'I-neutral': -29.991711633096948,
  'I-negative': -29.574300149101838},
 {'O': -24.870461504140945,
  'B-positive': -25.08080537736095,
  'B-negative': -24.95798314830894,
  'I-positive': -25.265509654950865,
  'B-neutral': -24.956464083760412,
  'I-neutral': -25.3774589934867,
  'I-negative': -25.2369862722523},
 {'O': -21.631883656823128,
  'B-positive': -4503599627370507.0,
  'B-negative': -4503599627370508.0,
  'I-positive': -4503599627370507.0,
  'B-neutral': -4503599627370508.0,
  'I-neutral': -4503599627370508.0,
  'I-negative': -4503599627370508.0},
 {'O': -11.373702903725697,
  'B-positive': -10.830678494159349,
  'B-negative': -11.461224547893693,
  'I-positive': -10.692627318251152,
  'B-neutral': -11.459705483345164,
  'I-neutral': -11.880700393071455,
  'I-negative': -11.74022767183705},
 {'O': -5.541292824797559,
  'B-positive': -5.7516366980175615,
  'B-negative': -5.628814468965554,
  'I-positive': -5.936340975607477,
  'B-neutral': -5.6272954044170245,
  'I-neutral': -6.048290314143316,
  'I-negative': -5.907817592908913},
 {'O': -2.5966054955940074,
  'B-positive': -4.252688120309395,
  'B-negative': -4.836281906951478,
  'I-positive': -4.564348191467836,
  'B-neutral': -3.245193133185574,
  'I-neutral': -4503599627370496,
  'I-negative': -4503599627370496}],
-84.87181815572558)
```

In [ ]:

```
def forward_backward_algorithm(sentence, count_y_dict, emission_transition_dict):
    feature_expectation = {}
    labels = count_y_dict.keys()
    fwd_pi, fwd_score = forward_algorithm(sentence, count_y_dict, emission_transition_dict)
    bkd_pi, bkd_score = backward_algorithm(sentence, count_y_dict, emission_transition_dict)

    # idx = 1
    for label in labels:
        string_transition = f"transition: {START_STATE_KEY}+{label}"
        string_emission = f"emission: {label}+{sentence[0][0]}"

        # update transition features
        update = bkd_pi[0][label] \
                + emission_transition_dict[string_transition] \
                + emission_transition_dict[string_emission] \
                - fwd_score

        feature_expectation[string_transition] = feature_expectation.get(string_transition,0)
+ np.exp(update)

        # update emission features
        update = fwd_pi[0][label] + bkd_pi[0][label] - fwd_score
        feature_expectation[string_emission] = feature_expectation.get(string_emission,0) + n
p.exp(update)

    # idx = 2 to N-1
    for idx in range(1,len(sentence)):
        for label in labels:
            string_emission = f"emission: {label}+{sentence[idx][0]}"

            # update transition features
            for prev_label in labels:
                string_transition = f"transition: {prev_label}+{label}"
                update = fwd_pi[idx-1][prev_label] \
                        + bkd_pi[idx][label] \
                        + emission_transition_dict[string_transition] \
                        + emission_transition_dict[string_emission] \
                        - fwd_score
                feature_expectation[string_transition] = feature_expectation.get(string_trans
ition,0) + np.exp(update)

            # update emission features
            update = fwd_pi[idx][label] + bkd_pi[idx][label] - fwd_score
            feature_expectation[string_emission] = feature_expectation.get(string_emission,0)
+ np.exp(update)

    # idx = N (STOP)
    idx = len(sentence)
    for label in labels:
        # update transition features
        string_transition = f"transition: {label}+{STOP_STATE_KEY}"
        update = fwd_pi[idx-1][label] + emission_transition_dict[string_transition] - fwd_sco
re
        feature_expectation[string_transition] = feature_expectation.get(string_transition,0)
+ np.exp(update)

    return feature_expectation

forward_backward_algorithm(train_sentences[0], count_y_dict, emission_transition_dict)
```

Out[ ]:

```
{'transition: START+O': 1.0000000000000284,
 'emission: O+All': 1.0000000000000284,
 'transition: START+B-positive': 0.0,
 'emission: B-positive+All': 0.0,
 'transition: START+B-negative': 0.0,
 'emission: B-negative+All': 0.0,
 'transition: START+I-positive': 0.0,
 'emission: I-positive+All': 0.0,
 'transition: START+B-neutral': 0.0,
 'emission: B-neutral+All': 0.0,
 'transition: START+I-neutral': 0.0,
 'emission: I-neutral+All': 0.0,
 'transition: START+I-negative': 0.0,
 'emission: I-negative+All': 0.0,
 'transition: O+O': 10.441907229302245,
 'transition: B-positive+O': 0.5428700135957362,
 'transition: B-negative+O': 1.185388749994202,
 'transition: I-positive+O': 0.0005068471541609425,
 'transition: B-neutral+O': 0.04993841289642202,
 'transition: I-neutral+O': 0.0,
 'transition: I-negative+O': 0.0,
 'emission: O+in': 1.0000000000000284,
 'transition: O+B-positive': 0.5433768607498974,
 'transition: B-positive+B-positive': 0.0,
 'transition: B-negative+B-positive': 0.0,
 'transition: I-positive+B-positive': 0.0,
 'transition: B-neutral+B-positive': 0.0,
 'transition: I-neutral+B-positive': 0.0,
 'transition: I-negative+B-positive': 0.0,
 'emission: B-positive+in': 0.0,
 'transition: O+B-negative': 1.1853887499942022,
 'transition: B-positive+B-negative': 0.0,
 'transition: B-negative+B-negative': 0.0,
 'transition: I-positive+B-negative': 0.0,
 'transition: B-neutral+B-negative': 0.0,
 'transition: I-neutral+B-negative': 0.0,
 'transition: I-negative+B-negative': 0.0,
 'emission: B-negative+in': 0.0,
 'transition: O+I-positive': 0.0,
 'transition: B-positive+I-positive': 0.0005068471541609574,
 'transition: B-negative+I-positive': 0.0,
 'transition: I-positive+I-positive': 0.00017787626272221284,
 'transition: B-neutral+I-positive': 0.0,
 'transition: I-neutral+I-positive': 0.0,
 'transition: I-negative+I-positive': 0.0,
 'emission: I-positive+in': 0.0,
 'transition: O+B-neutral': 0.04993841289642202,
 'transition: B-positive+B-neutral': 0.0,
 'transition: B-negative+B-neutral': 0.0,
 'transition: I-positive+B-neutral': 0.0,
 'transition: B-neutral+B-neutral': 0.0,
 'transition: I-neutral+B-neutral': 0.0,
 'transition: I-negative+B-neutral': 0.0,
 'emission: B-neutral+in': 0.0,
 'transition: O+I-neutral': 0.0,
 'transition: B-positive+I-neutral': 0.0,
 'transition: B-negative+I-neutral': 0.0,
 'transition: I-positive+I-neutral': 0.0,
 'transition: B-neutral+I-neutral': 0.0,
 'transition: I-neutral+I-neutral': 0.0,
 'transition: I-negative+I-neutral': 0.0,
 'emission: I-neutral+in': 0.0,
 'transition: O+I-negative': 0.0,
 'transition: B-positive+I-negative': 0.0,
```

```
'transition: B-negative+I-negative': 0.0,
'transition: I-positive+I-negative': 0.0,
'transition: B-neutral+I-negative': 0.0,
'transition: I-neutral+I-negative': 0.0,
'transition: I-negative+I-negative': 0.0,
'emission: I-negative+in': 0.0,
'emission: O+all': 0.9753657434645776,
'emission: B-positive+all': 0.011735572330503501,
'emission: B-negative+all': 0.012898684204926262,
'emission: I-positive+all': 0.0,
'emission: B-neutral+all': 0.0,
'emission: I-neutral+all': 0.0,
'emission: I-negative+all': 0.0,
'emission: O+,': 0.9994931528458538,
'emission: B-positive+,': 0.0,
'emission: B-negative+,': 0.0,
'emission: I-positive+,': 0.0005068471541609502,
'emission: B-neutral+,': 0.0,
'emission: I-neutral+,': 0.0,
'emission: I-negative+,': 0.0,
'emission: O+the': 1.9999064737555226,
'emission: B-positive+the': 0.0,
'emission: B-negative+the': 0.0,
'emission: I-positive+the': 9.352624449320648e-05,
'emission: B-neutral+the': 0.0,
'emission: I-neutral+the': 0.0,
'emission: I-negative+the': 0.0,
'emission: O+food': 0.24584588287669223,
'emission: B-positive+food': 0.5316412884193862,
'emission: B-negative+food': 0.17249006578927584,
'emission: I-positive+food': 8.435001822900635e-05,
'emission: B-neutral+food': 0.04993841289642202,
'emission: I-neutral+food': 0.0,
'emission: I-negative+food': 0.0,
'emission: O+was': 1.0,
'emission: B-positive+was': 0.0,
'emission: B-negative+was': 0.0,
'emission: I-positive+was': 0.0,
'emission: B-neutral+was': 0.0,
'emission: I-neutral+was': 0.0,
'emission: I-negative+was': 0.0,
'emission: O+great': 1.0,
'emission: B-positive+great': 0.0,
'emission: B-negative+great': 0.0,
'emission: I-positive+great': 0.0,
'emission: B-neutral+great': 0.0,
'emission: I-neutral+great': 0.0,
'emission: I-negative+great': 0.0,
'emission: O+(': 1.0,
'emission: B-positive+(': 0.0,
'emission: B-negative+(': 0.0,
'emission: I-positive+(': 0.0,
'emission: B-neutral+(': 0.0,
'emission: I-neutral+(': 0.0,
'emission: I-negative+(': 0.0,
'emission: O+except': 1.0,
'emission: B-positive+except': 0.0,
'emission: B-negative+except': 0.0,
'emission: I-positive+except': 0.0,
'emission: B-neutral+except': 0.0,
'emission: I-neutral+except': 0.0,
'emission: I-negative+except': 0.0,
'emission: O+for': 1.0,
'emission: B-positive+for': 0.0,
'emission: B-negative+for': 0.0,
```

```
 'emission: I-positive+for': 0.0,
 'emission: B-neutral+for': 0.0,
 'emission: I-neutral+for': 0.0,
 'emission: I-negative+for': 0.0,
 'emission: O+dessserts': 0.0,
 'emission: B-positive+dessserts': 0.0,
 'emission: B-negative+dessserts': 1.0,
 'emission: I-positive+dessserts': 0.0,
 'emission: B-neutral+dessserts': 0.0,
 'emission: I-neutral+dessserts': 0.0,
 'emission: I-negative+dessserts': 0.0,
 'emission: O+)': 1.0,
 'emission: B-positive+)': 0.0,
 'emission: B-negative+)': 0.0,
 'emission: I-positive+)': 0.0,
 'emission: B-neutral+)': 0.0,
 'emission: I-neutral+)': 0.0,
 'emission: I-negative+)': 0.0,
 'emission: O+.': 1.0,
 'emission: B-positive+.': 0.0,
 'emission: B-negative+.': 0.0,
 'emission: I-positive+.': 0.0,
 'emission: B-neutral+.': 0.0,
 'emission: I-neutral+.': 0.0,
 'emission: I-negative+.': 0.0,
 'transition: O+STOP': 1.0,
 'transition: B-positive+STOP': 0.0,
 'transition: B-negative+STOP': 0.0,
 'transition: I-positive+STOP': 0.0,
 'transition: B-neutral+STOP': 0.0,
 'transition: I-neutral+STOP': 0.0,
 'transition: I-negative+STOP': 0.0}
```

In [ ]:
```python
def get_features(sentences, count_y_dict, emission_transition_dict):
    features = {k:0 for k,v in emission_transition_dict.items()}
    for sent in sentences:
        expect = forward_backward_algorithm(sent, count_y_dict, emission_transition_dict)
        for k,v in expect.items():
            features[k] += v
    return features

features = get_features(train_sentences, count_y_dict, emission_transition_dict)
```

In [ ]:
```python
def mapping_fn(emission_transition_dict):
    index_map = {}
    for idx, value in enumerate(emission_transition_dict):
        if len(value.split(" ")[1].split("+")) > 2:
            first = str(value.split(" ")[1].split("+")[0])
            second = '+'
        else:
            first = str(value.split(" ")[1].split("+")[0])
            second = str(value.split(" ")[1].split("+")[1])
        index_map[idx] = (first,second)
    return index_map

index_map = mapping_fn(emission_transition_dict)
```

In [ ]:
```
def compute_grad(sentences, count_y_dict, features, index_map, emission_transition_dict):
    labels = count_y_dict.keys()
    counter = 0
    grad_lst = np.zeros(len(emission_transition_dict),)
    for i in range(len(features)):
        # print(index_map[i],count_y_to_x_dict.keys())
        if index_map[i] in count_y_to_x_dict.keys():
            string = f'emission: {index_map[i][0]}+{index_map[i][1]}'
            grad_lst[i] += (features[string] - count_y_to_x_dict[index_map[i]])
        elif index_map[i] in count_y_to_y_dict.keys():
            string = f'transition: {index_map[i][0]}+{index_map[i][1]}'
            grad_lst[i] += (features[string] - count_y_to_y_dict[index_map[i]])
        else:
            try:
                string = f'emission: {index_map[i][0]}+{index_map[i][1]}'
                grad_lst[i] += (features[string])
            except:
                pass
            try:
                string = f'transition: {index_map[i][0]}+{index_map[i][1]}'
                grad_lst[i] += (features[string])
            except:
                pass
    return grad_lst

compute_grad(train_sentences, count_y_dict, features, index_map, emission_transition_dict)
```

Out[ ]:
```
array([ 4.26325641e-14, -3.14611815e+00, -2.90336197e-01, ...,
        0.00000000e+00,  0.00000000e+00,  0.00000000e+00])
```

PART 4

4i)

In [ ]:
```
def loss_with_reg(w, sentences, count_y_dict, emission_transition_dict, n = 0.1):
    loss = loss_fn(sentences, count_y_dict, emission_transition_dict)
    loss += n*sum(w1*w1 for w1 in w)
    return loss
```

In [ ]:
```
def grad_with_reg(w, sentences, count_y_dict, features, index_map, emission_transition_dict,
n = 0.1):
    grad_lst = compute_grad(sentences, count_y_dict, features, index_map, emission_transition
_dict)
    for i in range(len(w)):
        grad_lst[i] += w[i]*2*n
    return grad_lst
```

In [ ]:

```
import time
from scipy.optimize import fmin_l_bfgs_b
total_start = time.time()

def callbackF(w):
    '''
    This function will only be called by "fmin_l_bfgs_b"
    Arg:
    w: weights, numpy array
    '''
    loss = get_loss_grad(w)[0]
    print('Loss:{0:.4f}'.format(loss))

def get_loss_grad(w):
    '''
    This function will be called by "fmin_l_bfgs_b"
    Arg:
    w: weights, numpy array
    Returns:
    loss: loss, float
    grads: gradients, numpy array
    '''
    start = time.time()
    new_emission_transition_dict = {}
    for i in range(len(index_map.keys())):
        if i < len(emission_dict):
            string = f'emission: {index_map[i][0]}+{index_map[i][1]}'
            new_emission_transition_dict[string] = w[i]
        else:
            string = f'transition: {index_map[i][0]}+{index_map[i][1]}'
            new_emission_transition_dict[string] = w[i]
    features = get_features(train_sentences, count_y_dict, new_emission_transition_dict)

    loss = loss_with_reg(w, train_sentences, count_y_dict, new_emission_transition_dict, n =
0.1)
    print('loss: '+ str(loss))

    grad_lst = grad_with_reg(w, train_sentences, count_y_dict, features, index_map, new_emiss
ion_transition_dict, n = 0.1)
    grads = np.asarray(list(grad_lst))

    print('time taken: '+ str(time.time()-start) +' total time: '+ str(time.time()-total_star
t))

    return loss, grads

init_w = np.zeros(len(index_map),)
results = fmin_l_bfgs_b(get_loss_grad, init_w, pgtol=0.01, maxiter=50, callback=callbackF)
```

```
loss: 52775.02915252912
time taken: 17.978718757629395 total time: 18.051718711853027
loss: 31978.422560595387
time taken: 18.113350868225098 total time: 36.192068576812744
loss: 19007.65777640147
time taken: 17.230767965316772 total time: 53.42583632469177
loss: 19007.65777640147
time taken: 16.818676471710205 total time: 70.24848580360413
Loss:19007.6578
loss: 15909.840926992103
time taken: 16.53831124305725 total time: 86.79179739952087
loss: 24692.805625405286
time taken: 16.614581823349 total time: 103.41038990020752
loss: 12068.697074799862
time taken: 16.62595796585083 total time: 120.0413224697113
loss: 12068.697074799862
time taken: 16.613444328308105 total time: 136.65776705741882
Loss:12068.6971
loss: 42879.194680706445
time taken: 16.625876426696777 total time: 153.28864431381226
loss: 11393.865776589319
time taken: 16.497926473617554 total time: 169.79057216644287
loss: 11393.865776589319
time taken: 16.60111403465271 total time: 186.3956868648529
Loss:11393.8658
loss: 10788.15567690617
time taken: 16.60189127922058 total time: 203.0015799999237
loss: 10788.15567690617
time taken: 17.085956573486328 total time: 220.09153628349304
Loss:10788.1557
loss: 10197.25111554726
time taken: 16.628634214401245 total time: 236.72517228126526
loss: 10197.25111554726
time taken: 16.60098910331726 total time: 253.32912611961365
Loss:10197.2511
loss: 9635.861130429555
time taken: 16.619665384292603 total time: 269.95379161834717
loss: 9635.861130429555
time taken: 16.644445657730103 total time: 286.6012647151947
Loss:9635.8611
loss: 8861.897153827294
time taken: 16.830179452896118 total time: 303.43744683265686
loss: 8861.897153827294
time taken: 16.622847318649292 total time: 320.0643198490143
Loss:8861.8972
loss: 8284.076747442528
time taken: 16.62486171722412 total time: 336.694149017334
loss: 8284.076747442528
time taken: 16.59043049812317 total time: 353.2885808944702
Loss:8284.0767
loss: 8078.550089725975
time taken: 16.590187788009644 total time: 369.885746717453
loss: 8078.550089725975
time taken: 16.5749409198761 total time: 386.4637129306793
Loss:8078.5501
loss: 7887.952221264266
time taken: 16.54378628730774 total time: 403.0124661922455
loss: 7887.952221264266
time taken: 16.708269357681274 total time: 419.7247359752655
Loss:7887.9522
loss: 7669.0498952222615
time taken: 16.537726879119873 total time: 436.2694640159607
loss: 7669.0498952222615
time taken: 16.644826412200928 total time: 452.91732358932495
```

```
Loss:7669.0499
loss: 7304.06179889031
time taken: 16.64111089706421 total time: 469.56443548202515
loss: 7304.06179889031
time taken: 16.589702606201172 total time: 486.1581389904022
Loss:7304.0618
loss: 6878.975796460469
time taken: 16.545958518981934 total time: 502.7100603580475
loss: 6878.975796460469
time taken: 16.674083948135376 total time: 519.3881425857544
Loss:6878.9758
loss: 6513.320422939939
time taken: 16.638832092285156 total time: 536.0339457988739
loss: 6513.320422939939
time taken: 16.643459796905518 total time: 552.6809771060944
Loss:6513.3204
loss: 6348.989212527013
time taken: 16.638678073883057 total time: 569.3256134986877
loss: 6348.989212527013
time taken: 16.60756492614746 total time: 585.9371800422668
Loss:6348.9892
loss: 6077.388244170685
time taken: 16.68115258216858 total time: 602.6243343353271
loss: 6077.388244170685
time taken: 17.22388982772827 total time: 619.8522250652313
Loss:6077.3882
loss: 5959.001193162048
time taken: 16.48738431930542 total time: 636.346287727356
loss: 5959.001193162048
time taken: 16.389054775238037 total time: 652.7393724918365
Loss:5959.0012
loss: 5657.876857057016
time taken: 16.28726887702942 total time: 669.0326397418976
loss: 5657.876857057016
time taken: 16.436951398849487 total time: 685.4725902080536
Loss:5657.8769
loss: 5434.752094427957
time taken: 16.318097591400146 total time: 701.7966890335083
loss: 5434.752094427957
time taken: 16.36106252670288 total time: 718.1617512702942
Loss:5434.7521
loss: 5743.379462404202
time taken: 16.333727598190308 total time: 734.5014815330505
loss: 5381.266331201675
time taken: 16.36314368247986 total time: 750.8686261177063
loss: 5381.266331201675
time taken: 16.37564444541931 total time: 767.2482380867004
Loss:5381.2663
loss: 5333.562398984037
time taken: 16.349736213684082 total time: 783.602972984314
loss: 5333.562398984037
time taken: 16.535582304000854 total time: 800.1425807476044
Loss:5333.5624
loss: 5302.8242387021355
time taken: 17.89575219154358 total time: 818.0453040599823
loss: 5302.8242387021355
time taken: 17.808614253997803 total time: 835.8579208850861
Loss:5302.8242
loss: 5214.821254215742
time taken: 17.9650776386261 total time: 853.8292245864868
loss: 5214.821254215742
time taken: 16.959732055664062 total time: 870.792956829071
Loss:5214.8213
loss: 5109.091252869708
time taken: 17.6808443069458 total time: 888.4797749519348
```

```
loss: 5109.091252869708
time taken: 17.473228454589844 total time: 905.9579770565033
Loss:5109.0913
loss: 4954.769118242758
time taken: 17.667571544647217 total time: 923.6315166950226
loss: 4954.769118242758
time taken: 18.016346216201782 total time: 941.6518633365631
Loss:4954.7691
loss: 4904.418471304348
time taken: 17.524449825286865 total time: 959.182285785675
loss: 4904.418471304348
time taken: 17.493335485458374 total time: 976.6795926094055
Loss:4904.4185
loss: 4758.848208067353
time taken: 17.150562286376953 total time: 993.8361518383026
loss: 4758.848208067353
time taken: 17.85442352294922 total time: 1011.6945745944977
Loss:4758.8482
loss: 4651.419723344664
time taken: 17.142289400100708 total time: 1028.842863559723
loss: 4651.419723344664
time taken: 17.079845428466797 total time: 1045.9267106056213
Loss:4651.4197
loss: 4557.89595774837
time taken: 17.435286045074463 total time: 1063.3679673671722
loss: 4557.89595774837
time taken: 17.7206974029541 total time: 1081.0926632881165
Loss:4557.8960
loss: 4489.40239521727
time taken: 17.577993154525757 total time: 1098.6766583919525
loss: 4489.40239521727
time taken: 18.121572494506836 total time: 1116.8022291660309
Loss:4489.4024
loss: 4391.653249200081
time taken: 17.33444905281067 total time: 1134.1426787376404
loss: 4391.653249200081
time taken: 17.335862398147583 total time: 1151.4825389385223
Loss:4391.6532
loss: 4344.1130161477495
time taken: 16.273280382156372 total time: 1167.7618191242218
loss: 4344.1130161477495
time taken: 18.005548238754272 total time: 1185.770367383957
Loss:4344.1130
loss: 4210.479992014277
time taken: 17.673574686050415 total time: 1203.4504790306091
loss: 4210.479992014277
time taken: 17.16085982322693 total time: 1220.6143424510956
Loss:4210.4800
loss: 4105.660003330031
time taken: 17.408498525619507 total time: 1238.0288400650024
loss: 4105.660003330031
time taken: 17.521984100341797 total time: 1255.5548276901245
Loss:4105.6600
loss: 4075.4750173385783
time taken: 17.691861867904663 total time: 1273.2526881694794
loss: 4075.4750173385783
time taken: 17.93489384651184 total time: 1291.1915826797485
Loss:4075.4750
loss: 4021.341228125867
time taken: 18.92495608329773 total time: 1310.12153673172
loss: 4021.341228125867
time taken: 18.017289876937866 total time: 1328.1428263187408
Loss:4021.3412
loss: 4128.563512919455
time taken: 17.753011226654053 total time: 1345.9018414020538
```

```
loss: 3972.4630536024843
time taken: 18.18455410003662 total time: 1364.0903975963593
loss: 3972.4630536024843
time taken: 18.343806266784668 total time: 1382.4372026920319
Loss:3972.4631
loss: 3984.8897487603012
time taken: 19.260965585708618 total time: 1401.7046530246735
loss: 3928.744925690231
time taken: 18.951504707336426 total time: 1420.6601302623749
loss: 3928.744925690231
time taken: 18.66208004951477 total time: 1439.327178478241
Loss:3928.7449
loss: 3883.9008003131125
time taken: 18.699676990509033 total time: 1458.0328559875488
loss: 3883.9008003131125
time taken: 17.547747373580933 total time: 1475.5855784416199
Loss:3883.9008
loss: 3868.378691175448
time taken: 18.83545184135437 total time: 1494.4265427589417
loss: 3868.378691175448
time taken: 18.17342448234558 total time: 1512.604968070984
Loss:3868.3787
loss: 3817.616628901539
time taken: 17.905985832214355 total time: 1530.5169517993927
loss: 3817.616628901539
time taken: 17.335237741470337 total time: 1547.856188774109
Loss:3817.6166
loss: 3794.772537462322
time taken: 16.96452522277832 total time: 1564.8256769180298
loss: 3794.772537462322
time taken: 17.465285062789917 total time: 1582.2959604263306
Loss:3794.7725
loss: 3737.794456893808
time taken: 17.78863024711609 total time: 1600.0905900001526
loss: 3737.794456893808
time taken: 18.916321992874146 total time: 1619.011886358261
Loss:3737.7945
loss: 3677.2696821612526
time taken: 17.659732818603516 total time: 1636.6766157150269
loss: 3677.2696821612526
time taken: 17.58705759048462 total time: 1654.2667014598846
Loss:3677.2697
loss: 3698.270810700038
time taken: 18.55267882347107 total time: 1672.8253817558289
loss: 3645.2906873179436
time taken: 17.03842782974243 total time: 1689.8677797317505
loss: 3645.2906873179436
time taken: 17.327402591705322 total time: 1707.199203968048
Loss:3645.2907
loss: 3629.341480729705
time taken: 18.741477012634277 total time: 1725.9470841884613
loss: 3629.341480729705
time taken: 17.57262897491455 total time: 1743.5237164497375
Loss:3629.3415
loss: 3590.4551955513252
time taken: 16.94676446914673 total time: 1760.476454257965
loss: 3590.4551955513252
time taken: 18.09673810005188 total time: 1778.5772321224213
Loss:3590.4552
loss: 3524.230616546091
time taken: 17.138217449188232 total time: 1795.7224493026733
loss: 3524.230616546091
time taken: 17.418748140335083 total time: 1813.1451964378357
Loss:3524.2306
loss: 3554.798747316445
```

```
time taken: 17.425973653793335 total time: 1830.5791726112366
loss: 3504.9608096835673
time taken: 16.96469521522522 total time: 1847.5478699207306
loss: 3504.9608096835673
time taken: 17.256277084350586 total time: 1864.807143688202
Loss:3504.9608
loss: 3481.1155723787915
time taken: 18.75818133354187 total time: 1883.5713241100311
loss: 3481.1155723787915
time taken: 18.832395553588867 total time: 1902.407747745514
Loss:3481.1156
```

4ii)

In [ ]:
```
new_emission_transition_dict = {}
for i in range(len(index_map.keys())): #27899
    if i< len(emission_dict): #27818
        string = f'emission: {index_map[i][0]}+{index_map[i][1]}'
        new_emission_transition_dict[string] = results[0][i]
    else:
        string = f'transition: {index_map[i][0]}+{index_map[i][1]}'
        new_emission_transition_dict[string] = results[0][i]

viterbi_algo(test_sentences, count_y_dict, new_emission_transition_dict, 'dev.p4.out')
```

In [ ]:
```
prediction_dir = 'output/dev.p4.out'
truth_dir = 'dataset/dev.out'

lines = evaluate_results(truth_dir,prediction_dir)
res = conlleval.evaluate(lines)
print(conlleval.report(res))
```

```
processed 3809 tokens with 210 phrases; found: 153 phrases; correct: 66.
accuracy:  92.23%; precision:  43.14%; recall:  31.43%; FB1:  36.36
         negative: precision:  50.00%; recall:   9.23%; FB1:  15.58  12
          neutral: precision:   0.00%; recall:   0.00%; FB1:   0.00  3
         positive: precision:  43.48%; recall:  43.80%; FB1:  43.64  138
```

Part 5

In [ ]:

```
def unigram_1_parameters(train_dir, emission_dict):
    """Calculates the transition parameters by count(y->x_i-1)/count(y)

    :param train_dir: our train file
    :type train_sentences: str

    :param emission_dict: count(y->x_i-1)/count(y), keys are tuples of word and label ('unigr
am_1: O+All', -9.01768561), value MLE
    :type emission_dict: dict()

    :return count_y_to_y_dict: Count of labels and previous label
    :rtype: dict()

    :return emission_transition_dict: value of Count(labels->words_i-1)/Count(labels) for emi
ssion and Count(prev_labels->labels)/Count(labels) for transmission, keys are tuples of word
 and label ('unigram_1: O+All', -9.01768561), value MLE
    :rtype: dict()
    """
    # key is label | value is count
    count_y_dict = {}
    # key is word_i-1 , label_i | value is count
    count_y_to_x_dict = {}

    with open(train_dir, "r", encoding="utf8") as f:
        prev_word, prev_label = "", ""
        for line in f:
            # Parse each line
            if len(line.split(" ")) == 2:
                word, label = line.replace("\n", "").split(" ")
            else:
                label = ""

            # counting
            if label == "" and prev_label != "":
                count_y_dict[STOP_STATE_KEY] = count_y_dict.get(STOP_STATE_KEY, 0) + 1

            elif label != "":
                if prev_label == "":
                    count_y_dict[START_STATE_KEY] = (
                        count_y_dict.get(START_STATE_KEY, 0) + 1
                    )
                if label in count_y_dict:
                    count_y_dict[label] = count_y_dict.get(label) + 1
                else:
                    count_y_dict[label] = 1

            # Counting unigram
            if label != "" and prev_word != "":
                count_y_to_x_dict[(label, prev_word)] = (
                    count_y_to_x_dict.get((label, prev_word), 0) + 1
                )

            prev_word, prev_label = word, label

    # Calculate unigram
    for key, value in count_y_to_x_dict.items():  # Default is iterate keys()
        label = key[0]
        word = key[1]
        string = f"unigram_1: {label}+{word}"

        prob = value / count_y_dict.get(label)
        emission_dict[string] = float(np.where(prob != 0, np.log(prob), LARGE_NEG))

    print(
```

```
        "unigram_1 yi -> xi-1: \n",
        list(emission_dict.items())[-10:],
        len(emission_dict),
        "\n",
    )
    emission_transition_dict = emission_dict

    return count_y_to_x_dict, emission_transition_dict
```

In [ ]:

```
def unigram_2_parameters(train_dir, emission_dict):
    """Calculates the transition parameters by count(y->x_i+1)/count(y)

    :param train_dir: our train file
    :type train_sentences: str

    :param emission_dict: count(y->x_i+1)/count(y), keys are tuples of word and label ('unigr
am_1: O+All', -9.01768561), value MLE
    :type emission_dict: dict()

    :return count_y_to_y_dict: Count of labels and previous label
    :rtype: dict()

    :return emission_transition_dict: value of Count(labels -> words_i+1)/Count(labels) for e
mission and Count(prev_labels->labels)/Count(labels) for transmission, keys are tuples of wor
d and label ('unigram_1: O+All', -9.01768561), value MLE
    :rtype: dict()
    """
    # key is label | value is count
    count_y_dict = {}
    # key is word_i+1 , label_i | value is count
    count_y_to_x_dict = {}

    with open(train_dir, "r", encoding="utf8") as f:
        prev_word, prev_label = "", ""
        for line in f:
            # Parse each line
            if len(line.split(" ")) == 2:
                word, label = line.replace("\n", "").split(" ")
            else:
                label = ""

            # counting
            if label == "" and prev_label != "":
                count_y_dict[STOP_STATE_KEY] = count_y_dict.get(STOP_STATE_KEY, 0) + 1
            elif label != "":
                if prev_label == "":
                    count_y_dict[START_STATE_KEY] = (
                        count_y_dict.get(START_STATE_KEY, 0) + 1
                    )
                if label in count_y_dict:
                    count_y_dict[label] = count_y_dict.get(label) + 1
                else:
                    count_y_dict[label] = 1

            if prev_label != "" and word != "":
                count_y_to_x_dict[(prev_label, word)] = (
                    count_y_to_x_dict.get((prev_label, word), 0) + 1
                )

            prev_word, prev_label = word, label

    # Calculate unigram
    for (label, word), value in count_y_to_x_dict.items():  # Default is iterate keys()
        if prev_label != "" and word != "":
            string = f"unigram_2: {label}+{word}"
            prob = value / count_y_dict.get(label)
            emission_dict[string] = float(np.where(prob != 0, np.log(prob), LARGE_NEG))

    print(
        "unigram_2 yi -> x_i+1: \n",
        list(emission_dict.items())[-10:],
        len(emission_dict),
        "\n",
```

```
    )
    emission_transition_dict = emission_dict

    return count_y_to_x_dict, emission_transition_dict
```

In [ ]:

```
def bigram_parameters(train_dir, emission_dict):
    """Calculates the transition parameters by count(y->x_i+1)/count(y)

    :param train_dir: our train file
    :type train_sentences: str

    :param emission_dict: count(yi-1 -> yi -> xi)/count(y), keys are tuples of word and label
('B-neutral+O+B-neutral', -9.01768561)
    :type emission_dict: dict()

    :return count_y_to_y_dict: Count of labels and previous label
    :rtype: dict()

    :return emission_transition_dict: value of Count(label-1 -> labels -> words)/Count(label
s) for emission and Count(prev_labels->labels)/Count(labels) for transmission, keys are tuple
s of word and label ('B-neutral+O+B-neutral', -9.01768561)
    :rtype: dict()
    """
    # key is label | value is count
    count_y_dict = {}
    # key is word_i+1 , label_i | value is count
    count_y_to_y_to_x_dict = {}

    with open(train_dir, "r", encoding="utf8") as f:
        prev_word, prev_label = "", ""
        for line in f:
            # Parse each line
            if len(line.split(" ")) == 2:
                word, label = line.replace("\n", "").split(" ")
            else:
                label = ""

            # counting
            if label == "" and prev_label != "":
                count_y_dict[STOP_STATE_KEY] = count_y_dict.get(STOP_STATE_KEY, 0) + 1
            elif label != "":
                if prev_label == "":
                    count_y_dict[START_STATE_KEY] = (
                        count_y_dict.get(START_STATE_KEY, 0) + 1
                    )
                if label in count_y_dict:
                    count_y_dict[label] = count_y_dict.get(label) + 1
                else:
                    count_y_dict[label] = 1

            if prev_label != "" and word != "" and label != "":
                count_y_to_y_to_x_dict[(prev_label, label, word)] = (
                    count_y_to_y_to_x_dict.get((prev_label, label, word), 0) + 1
                )

            prev_label = label

    # Calculate unigram
    for key, value in count_y_to_y_to_x_dict.items():  # Default is iterate keys()
        prev_label, label, word = key
        if prev_label != "" and label != "" and word != "":
            string = f"bigram: {prev_label}+{label}+{word}"
            prob = value / count_y_dict.get(label)
            emission_dict[string] = float(np.where(prob != 0, np.log(prob), LARGE_NEG))
        prev_label = label

    print(
        "bigram yi-1 -> yi -> xi: \n",
        list(emission_dict.items())[-10:],
```

```
        len(emission_dict),
        "\n",
    )
    emission_transition_dict = emission_dict

    return count_y_to_y_to_x_dict, emission_transition_dict
```

In [ ]:
```
_, _, emission_dict = MLE_emission_parameters(train_sentences)
_, emission_dict = MLE_transition_parameters(train_dir, emission_dict)
_, emission_dict = unigram_1_parameters(train_dir, emission_dict)
_, emission_dict = unigram_2_parameters(train_dir, emission_dict)
count_y_dict, emission_dict = bigram_parameters(train_dir, emission_dict)

print(list(emission_dict.items())[:5])
print(list(emission_dict.items())[-5:])
```

```
unigram_1 yi -> xi-1:
 [('unigram_1: O+combination', -10.116297899710545), ('unigram_1: B-positive+super-fresh', -
7.0859014643656115), ('unigram_1: O+unusual', -10.116297899710545), ('unigram_1: B-neutral+bi
ggest', -4.343805421853684), ('unigram_1: O+adequate', -10.116297899710545), ("unigram_1: O+M
om's", -10.116297899710545), ('unigram_1: O+leaving', -10.116297899710545), ("unigram_1: O+w
e're", -10.116297899710545), ('unigram_1: O+hurry', -10.116297899710545), ('unigram_1: O+orig
inally', -10.116297899710545)] 32447

unigram_2 yi -> x_i+1:
 [('unigram_1: O+combination', -10.116297899710545), ('unigram_1: B-positive+super-fresh', -
7.0859014643656115), ('unigram_1: O+unusual', -10.116297899710545), ('unigram_1: B-neutral+bi
ggest', -4.343805421853684), ('unigram_1: O+adequate', -10.116297899710545), ("unigram_1: O+M
om's", -10.116297899710545), ('unigram_1: O+leaving', -10.116297899710545), ("unigram_1: O+w
e're", -10.116297899710545), ('unigram_1: O+hurry', -10.116297899710545), ('unigram_1: O+orig
inally', -10.116297899710545)] 32447

bigram yi-1 -> yi -> xi:
 [('bigram: O+O+combination', -10.116297899710545), ('bigram: O+O+super-fresh', -10.116297899
710545), ('bigram: O+O+unusual', -10.116297899710545), ('bigram: O+O+biggest', -10.1162978997
10545), ('bigram: O+O+adequate', -10.116297899710545), ("bigram: O+O+Mom's", -10.116297899710
545), ('bigram: O+O+leaving', -10.116297899710545), ("bigram: O+O+we're", -10.11629789971054
5), ('bigram: O+O+hurry', -10.116297899710545), ('bigram: O+O+originally', -10.11629789971054
5)] 37068

[('emission: O+All', -9.017685611042436), ('emission: O+in', -4.54034879656423), ('emission:
O+all', -5.785564559424215), ('emission: O+,', -3.24728344904484), ('emission: O+the', -3.091
6488692569097)]
[("bigram: O+O+Mom's", -10.116297899710545), ('bigram: O+O+leaving', -10.116297899710545),
("bigram: O+O+we're", -10.116297899710545), ('bigram: O+O+hurry', -10.116297899710545), ('big
ram: O+O+originally', -10.116297899710545)]
```

In [ ]:

```
def viterbi_algo_2(test_sentences, count_y_dict, emission_dict):
    """Decoding process that finds greedily finds the best possible labels from past MLE scor
es, saves file to output folder

    :param test_sentences: our file tokenised sentences
    :type test_sentences: list(tuple())

    :param count_y_dict: Count of labels
    :param count_y_dict: dict()

    :param emission_dict: value of Count(labels->words)/Count(labels) for emission and Count
(prev_labels->labels)/Count(labels) for transmission, keys are tuples of word and label ('emi
ssion: O+All', -9.01768561), value MLE
    :param emission_dict: dict()
    """

    pi = [{}]
    path = {}
    labels = count_y_dict.keys()
    os.makedirs("output", exist_ok=True)

    with open("output/dev.p5.out", "w") as outfile:
        for sentence in test_sentences:
            # j = 0 (START)
            for label in labels:
                pi[0][label] = emission_dict.get(
                    f"transition: {'START'}+{label}", LARGE_NEG
                ) + emission_dict.get(f"emission: {label}+{sentence[0][0]}", LARGE_NEG)
                path[label] = [label]
            # j = 1 to N-1
            for idx in range(1, len(sentence)):
                pi.append({})
                newpath = {}
                for label_y in labels:
                    (prob, label) = max(
                        [
                            (
                                pi[idx - 1][prev_label]
                                + emission_dict.get(
                                    f"transition: {prev_label}+{label_y}", LARGE_NEG
                                )
                                + emission_dict.get(
                                    f"emission: {label_y}+{sentence[idx][0]}", LARGE_NEG
                                )
                                + (
                                    emission_dict.get(
                                        f"unigram_1: {label_y}+{sentence[idx-1][0]}",
                                        LARGE_NEG,
                                    )
                                )
                                + (
                                    emission_dict.get(
                                        f"unigram_2: {label_y}+{sentence[idx+1][0]}",
                                        LARGE_NEG,
                                    )
                                    if idx < len(sentence) - 1
                                    else 0
                                )
                                + emission_dict.get(
                                    f"bigram: {prev_label}+{label_y}+{sentence[idx][0]}",
                                    LARGE_NEG,
                                ),
                                prev_label,
                            )
```

```
                    for prev_label in labels
                ]
            )
            pi[idx][label_y] = prob
            newpath[label_y] = path[label] + [label_y]
    path = newpath
# j = N (STOP)
idx = len(sentence)
(prob, label) = max(
    [
        (
            pi[idx - 1][label_y]
            + emission_dict.get(
                f"transition: {label_y}+{'STOP'}", LARGE_NEG
            ),
            label_y,
        )
        for label_y in labels
    ]
)

# handle inconsistent length
if len(sentence) != len(path[label]):
    print(len(sentence), len(path[label]))
    raise Exception(
        "{} has a different lenght with {}".format(sentence, path[label])
    )

# write to file
for i in range(len(sentence)):
    line = f"{sentence[i][0]} {path[label][i]}\n"
    outfile.write(line)

outfile.write("\n")
```

In [ ]:
```
viterbi_algo_2(test_sentences, count_y_dict, emission_dict)

lines = evaluate_results("dataset/dev.out", "output/dev.p5.out")
res = conlleval.evaluate(lines)
print(conlleval.report(res))
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-55-2f746ad0a298> in <module>
----> 1 viterbi_algo_2(test_sentences, count_y_dict, emission_dict)
      2
      3 lines = evaluate_results("dataset/dev.out", "output/dev.p5.out")
      4 res = conlleval.evaluate(lines)
      5 print(conlleval.report(res))

<ipython-input-54-dd2eec5fa8f3> in viterbi_algo_2(test_sentences, count_y_dict, emission_dic
t)
     60                                 prev_label,
     61                             )
---> 62                             for prev_label in labels
     63                         ]
     64                     )

<ipython-input-54-dd2eec5fa8f3> in <listcomp>(.0)
     60                                 prev_label,
     61                             )
---> 62                             for prev_label in labels
     63                         ]
     64                     )

KeyboardInterrupt:
```