Least frequent Words :
[['bosak', 6],
['caixa', 6],
['mapco', 6],
['ottis', 6],
['troup', 6],
['ccair', 7],
['cleft', 7],
['fabri', 7],
['foamy', 7],
['niaid', 7],
['paxon', 7],
['serna', 7],
['tocor', 7],
['yalom', 7],
['bitty', 8]]

Most Frequent Words:
[['sixty', 73086],
['there', 86502],
['years', 88900],
['forty', 94951],
['other', 106052],
['fifty', 106869],
['first', 109957],
['after', 110102],
['which', 142146],
['their', 145434],
['about', 157448],
['would', 159875],
['eight', 165764],
['seven', 178842],
['three', 273077]]

```
################################### Source ###################################
import os;
import math;

os.chdir('D:\\UCSD\\F16\\sduc-AI\\CSE250A\\HW\\HW1')
wordLength =5;

#Read the input file.
lines = [line.rstrip('\n') for line in open('hw1_word_counts_05.txt')]
totalWords = len(lines)
wordCount = []

for i in range (totalWords):
    itemSplit = lines[i].split(' ')
    itemSplit[0] = str(itemSplit[0]).lower()
    itemSplit[1] = int(itemSplit[1])
    wordCount.append(itemSplit)

wordCount.sort(key= lambda wordCount : wordCount[1]);

print ("Least frequent Words : ", wordCount[0:5]) #minimum used
print ("Most Frequent Words :", wordCount[totalWords-5:totalWords]) #maximum used
#AlphaLE denotes the Evidence information for predicted words.
#AlphaLE[i][j] > 0  => At position i, ('a' + Alpha[i][j] -1)th character predcited to appear.
#AlphaLE[i][j] < 0  => At position i, ('a' - Alpha[i][j] -1)th character predcited not to appear.

#Case#1
#AlphaLE = [[] for i in range (wordLength)]

#Case#2
#AlphaLE = [[-1, -9] for i in range (wordLength)]

#Case#3
#AlphaLE = [[-1, -18] for i in range (wordLength)]
#AlphaLE[0] = [1]
```

```
#AlphaLE[4] = [-1, 18]

#Case#4
#AlphaLE = [[-5, -1, -18] for i in range (wordLength)]
#AlphaLE[0] = [-5, 1]
#AlphaLE[4] = [-5, -1, 18]

#Case#5
#AlphaLE = [[-15, -4, -12, -3, -21] for i in range (wordLength)]
#AlphaLE[2] = [-15, -4, -12, -3, 21]

#Case6
#AlphaLE = [[-15,-5] for i in range (wordLength)]

#Case7
#AlphaLE = [[-4,-9] for i in range (wordLength)]
#AlphaLE[0] = [4]
#AlphaLE[3] = [-4, 9]

#Case#8
#AlphaLE = [[-1, -4, -9] for i in range (wordLength)]
#AlphaLE[0] = [-1,4]
#AlphaLE[3] = [-1, -4, 9]

#Case9
#AlphaLE = [[-1,-4,-9] for i in range (wordLength)]
#AlphaLE[0] = [-1, 4]
#AlphaLE[3] = [-1, -4, 9]

#Case10
#AlphaLE = [[-1, -5, -9, -15, -19, -21] for i in range (wordLength)]
#AlphaLE[1] = [-1, -5, -9, -15, -19, 21]

AlphaP = [0 for i in range(26)]

WordP = [(wordCount[i][1]/totalWords)for i in range(totalWords)]

def ProLEGivenW (wordIndex):
    p = 1;
    for i in range (len(AlphaLE)):
        if (AlphaLE[i] == []):      # if nothing predicted, every word id eligible.
            p = p*1;
        else :
            for j in range (len(AlphaLE[i])):
                if (AlphaLE[i][j] > 0 and wordCount[wordIndex][0][i] == chr(AlphaLE[i][j] -1 + ord('a'))):
                    p = p*1;
                elif(AlphaLE[i][j] < 0 and wordCount[wordIndex][0][i] != chr(-AlphaLE[i][j] -1 + ord('a'))):
                    p = p*1;
                else:
                    p = p*0;
                    return p ;
    #print ((wordCount[wordIndex]))
```

```
    return p;

def predictiveProbability():
        denominator = 0;
        #denominator is independent of summation over all words (outside.)
        for i in range (totalWords):
                denominator = denominator + (ProLEGivenW(i) * WordP[i]);
        for l in range (26):
                pdp = 0;
                for w in range (totalWords):
                        prob = 1;
                        flag = 0;
                        for i in range (len(AlphaLE)):
          #taking only poisitions which are not predicted yet, or not currently predicted.
                                if (AlphaLE[i] == [] or ( len(AlphaLE[i]) > 0 and
AlphaLE[i][len(AlphaLE[i])-1] < 0)) :
                                        if (wordCount[w][0][i] == chr(l + ord('a'))):
                                                prob = prob*1;
                                                flag = 1;
                        prob = prob * flag;
                        if (prob == 1):
                                prob = prob * ProLEGivenW(w) * WordP[w];
                        pdp = pdp + prob;
                AlphaP[l] = pdp/denominator;

def predict():
    predictiveProbability()
    maxAlpha = max(AlphaP)
    print (maxAlpha)
    for i in range(26):
        if (AlphaP[i] == maxAlpha):
            print (chr(ord('a') + i))

predict()
```