4.3

a. MILLION: 0.00207317938258646

MORE: 0.0017092454449816254

MR.: 0.0014419006093003488

MOST: 0.00078807703930133

MARKET: 0.0007805294866307582

MAY: 0.0007300452891081488

M.: 0.0007035493424558658

MANY: 0.0006968703088122154

MADE: 0.0005599745844420866

MUCH: 0.0005147015010810064

MAKE: 0.0005145669417951454

MONTH: 0.00044499979100496375

MONEY: 0.00043719535242502073

MONTHS: 0.0004058430388193876

MY: 0.00040039950407318907

MONDAY: 0.0003820627432090283

MAJOR: 0.0003709677184566642

MILITARY: 0.0003521171857737611

MEMBERS: 0.0003361290960809938

MIGHT: 0.0002736446567920457

MEETING: 0.0002657912875626987

MUST: 0.00026656194529081216

ME: 0.0002636261063265702 6

MARCH: 0.00025984621366010886

MAN: 0.0002529347594317894

MS.: 0.00023903845500104447

MINISTER: 0.000239821345391509

MAKING: 0.00021174738529594595

MOVE: 0.00020999811457975183

MILES: 0.00020601026665332327

b. THE <UNK> probability: 0.6150223623792234

THE U. probability: 0.013372554929385305

THE FIRST probability: 0.011720309314904268

THE COMPANY probability: 0.011658836440393748

THE NEW probability: 0.009451519300796631

THE UNITED probability: 0.00867234413189537

THE GOVERNMENT probability: 0.006803516870932123

THE NINETEEN probability: 0.006650742511916529

THE SAME probability: 0.006287092849200295

THE TWO probabilities: 0.006160775170353785

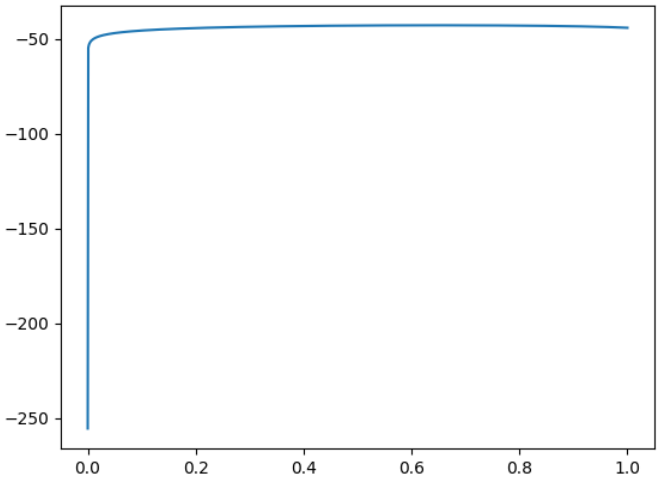
c. THE STOCK MARKET FELL BY ONE HUNDRED POINTS LAST WEEK
Unigram log likelihood sentence: -64.50741322999072
Bigram log likelihood sentence: -40.91813213378977

The bigram model yields the highest log likelihood.

d. THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE
Unigram log likelihood sentence: -44.29071820493777
Bigram log likelihood sentence: -infinity (zero probability element)

Tuples contributing to zero probability:
SIXTEEN - OFFICIALS
SOLD – FIRE

The log likelihood becomes -infinity.



e.

Optimal value of lambda: 0.648

4.4 Coefficients: [0.95067337 0.01560133 0.03189569]

a. Mean Squared Error on Nasdaq00: 13902.4010764
b. Mean Squared Error on Nasdaq01: 2985.09792411

Since the value of MSE is reducing, this means that we have not overfit the data but it could be an unknown fit due to less amount of data. Hence, we cannot use this model for accurate predictions.

Code for 4.3:

```
from math import log
from matplotlib import pyplot as plt

eps = 1e-50

# Reading words from vocab file
words = []
f = open('hw4_vocab.txt')
for word in f.read().split():
    words.append(word)
count = len(words)

# Reading counts from unigram file
ucount = []
f = open('hw4_unigram.txt')
for word in f.read().split():
    ucount.append(int(word))

# Calculating max likelihood estimate of words in unigram modelszszs
total = 0;
for i in range(0, len(words)-1):
    #print(words[i] + "-" + str(ucount[i]))
    total+=ucount[i]
uml = []
for i in range(0, len(words)-1):
    uml.append(ucount[i]/total);
    if(words[i].startswith('M')):
```

```
        print(words[i] + " : " + str(uml[i]))

#Reading from the bigram file

bicount = [[0 for x in range(count)] for y in range(count)]
temp = []
f = open('hw4_bigram.txt')
for word in f.read().split():
    temp.append(int(word))
print(len(temp))
for i in range(0, len(temp)-1, 3):
    bicount[temp[i]-1][temp[i+1]-1] = temp[i+2]

#Calculating probabilities of words most likely to occur after word 'THE"

the_posn = words.index('THE')
print(words[the_posn])

print(bicount[the_posn])
count_the = 0;
for i in range(0, len(bicount[the_posn])-1):
    count_the += bicount[the_posn][i]
print("Bigram count for the: " +str(count_the))
print("Unigram count for the: " + str(ucount[the_posn]))

sortedposns = sorted(range(len(bicount[the_posn])), key=lambda k:
bicount[the_posn][k])
print(sortedposns)
print("Hello")
for i in range (len(sortedposns)-1, len(sortedposns)-11, -1):
    #print(i)
    print("THE " + words[sortedposns[i]] + " probability : " +
str(bicount[the_posn][sortedposns[i]]/count_the))

#Using both models to calculate log likelihoods of the sentences

sentence1 = "THE STOCK MARKET FELL BY ONE HUNDRED POINTS LAST WEEK"
sentence2 = "THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE"

def loglikhelihood(sentence):
    print(sentence)
    w1 = sentence.split(' ')
    ull = 0;
    bll = 0;
    for w in w1:
        index = words.index(w)
        prob = ucount[index] / total
        ull += log(prob)

    print("Unigram log likelihood sentence: " + str(ull));

    startIndex = words.index('<s>')
    firstWordIndex = words.index(w1[0])
    bll += log(bicount[startIndex][firstWordIndex] / sum(bicount[startIndex]))

    for i in range(0, len(w1) - 1):
        curr = words.index(w1[i])
        nxt = words.index(w1[i + 1])
        prob = bicount[curr][nxt]/sum(bicount[curr])
        if(prob < eps):
            print(words[curr]+ " - " +words[nxt])
            prob = eps
        bll += log(prob)
```

```
    print("Bigram log likelihood sentence: " + str(bll))

loglikhelihood(sentence1)
loglikhelihood(sentence2)

def mixtureProb(l, uniprob, biprob):
    return (l*(uniprob)+(1-l)*(biprob))

def weightedmode(sentence):
    print(sentence)
    w1 = sentence.split(' ')
    uniprob = []
    biprob = []
    lam=0.0
    startIndex = words.index('<s>')
    firstWordIndex = words.index(w1[0])
    biprob.append(bicount[startIndex][firstWordIndex]/sum(bicount[startIndex]))
    for i in range(0, len(w1)):
        curr = words.index(w1[i])
        if(i+1 < len(w1)):
            print(w1[i])
            print(w1[i + 1] + "--")
            nxt = words.index(w1[i + 1])
            uniprob.append(ucount[curr] / total)
            biprob.append(bicount[curr][nxt] / sum(bicount[curr]))
        else:
            uniprob.append(ucount[curr]/total)

    X = []
    Y = []
    for l in range(0, 1001):
        print(lam)
        X.append(lam)
        ll = 0
        for i in range(0, len(biprob)):
            prob = mixtureProb(lam, uniprob[i], biprob[i])
            print(prob)
            if(prob<eps):
                ll+=log(eps)
            else:
                ll += log(prob)
        print(str(ll)+'log likelihood');
        Y.append(ll)
        lam += 0.001
    max_value = max(Y)
    print(max_value)
    max_index = Y.index(max_value)
    print(X[max_index])
    plt.plot(X, Y)
    plt.show()

weightedmode(sentence2)
```

Code for 4.4:

```
import numpy as np
from numpy.linalg import inv

data00 = []
data01 = []
```

```
f = open('hw4_nasdaq00.txt')
for word in f.read().split():
    data00.append(float(word))
f = open('hw4_nasdaq01.txt')
for word in f.read().split():
    data01.append(float(word))

data00count = len(data00)

X  = [[0 for x in range(3)] for y in range(data00count)]
Y = data00

for i in range(0, data00count):
    k=1
    while(k<=3):
        if(i-k>=0):
            X[i][k-1] = Y[i-k]
        k = k+1
X = X[3:]
Y = Y[3:]

theta, residuals, rank, s = np.linalg.lstsq(X, Y)
print(theta)
Xt = np.transpose(X)
A = np.matmul(Xt, X)
print(A)

Ainv = inv(A)
B = np.matmul(Xt, Y)
print(B)

Weight = np.matmul(Ainv, B)
Wt = np.transpose(Weight)

Ycalc = np.matmul(X, Wt)

MSE = ((Y - Ycalc)**2).mean()

print(Weight)
print("Mean Squared Error on Nasdaq00: " +str(MSE))

data01count = len(data01)
X  = [[0 for x in range(3)] for y in range(data01count)]
Y = data01

for i in range(0, data01count):
    k=1
    while(k<=3):
        if(i-k>=0):
            X[i][k-1] = Y[i-k]
        k = k+1

X=X[3:]
Y = Y[3:]
Ycalc = np.matmul(X, Wt)
MSE = ((Y - Ycalc)**2).mean()
print("Mean Squared Error on Nasdaq01: " +str(MSE))
```