(a)
for word BY,   the probality is 0.004180
for word BE,   the probality is 0.003370
for word BUT,   the probality is 0.002994
for word BEEN,   the probality is 0.001658
for word BECAUSE, the probality is 0.000902
for word BILLION, the probality is 0.000822
for word B.,   the probality is 0.000684
for word BEFORE, the probality is 0.000666
for word BUSINESS,the probality is 0.000541
for word BUSH,   the probality is 0.000516
for word BANK,   the probality is 0.000464
for word BETWEEN, the probality is 0.000458
for word BEING,   the probality is 0.000453
for word BACK,   the probality is 0.000448
for word BASED,   the probality is 0.000424
for word BOTH,   the probality is 0.000400
for word BIG,   the probality is 0.000340
for word BOARD,   the probality is 0.000334
for word BEGAN,   the probality is 0.000276
for word BILL,   the probality is 0.000267
for word BLACK,   the probality is 0.000239
for word BONDS,   the probality is 0.000213
for word BEST,   the probality is 0.000212
for word BETTER, the probality is 0.000212
for word BUY,   the probality is 0.000209
for word BUDGET, the probality is 0.000205
for word BANKS,   the probality is 0.000201

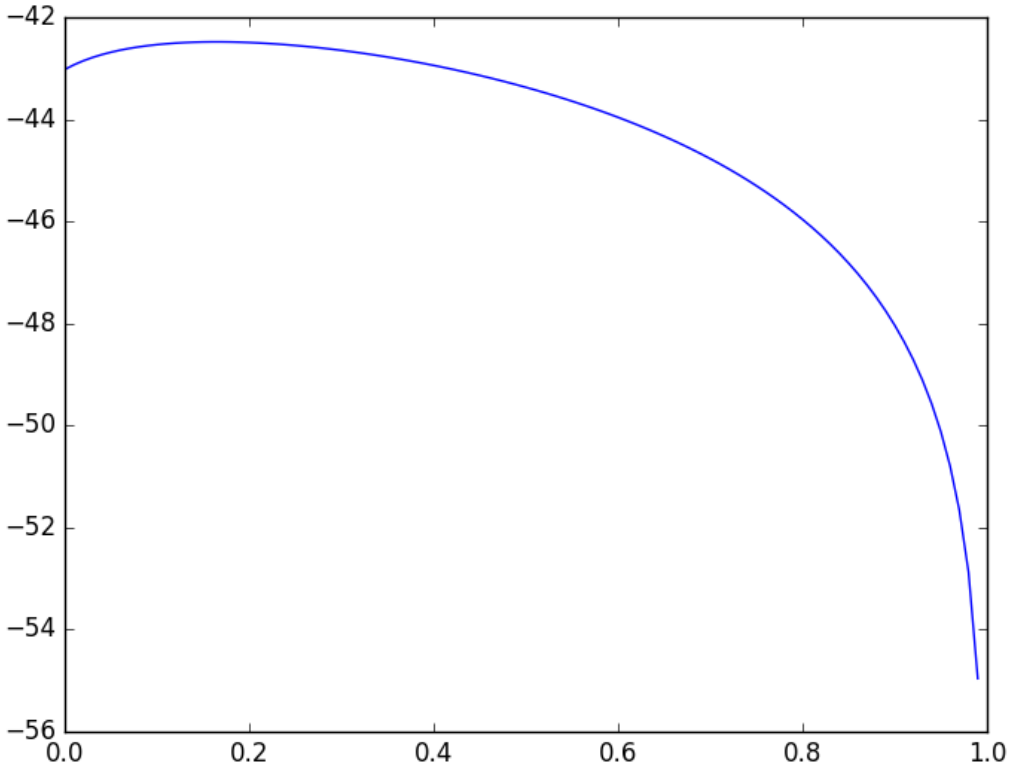
(b)
The word HUNDRED has probality 0.209061
The word <UNK>   has probality 0.124304
The word .POINT   has probality 0.099952
The word OF   has probality 0.073947
The word THOUSAND has probality 0.068654
The word MILLION has probality 0.031832
The word ,COMMA   has probality 0.031622
The word -HYPHEN has probality 0.030479
The word HALF   has probality 0.029139
The word .PERIOD has probality 0.024376

(c)
Lu = -64.509
Lb = -40.918
The Lb is higher than Lu

(d)
Lu = -44.2364299857
Lb = undefined,
no FOURTEEN followed by OFFICIALS

So the missing adjacent pair will cause the undefined problem in Lb

(e)



The highest should around 0.2

Codefrom math import log
import matplotlib.pyplot as plt

```
def lu(judge):
        token = judge.upper().strip('\n').split(' ')
        p = 1.0
        for i in range(0,len(token)):
                p *= cal1(token[i])
        return log(p*1.0)

def cal1(val1):
        return tokenDict[val1]*1.0/ totalCount

def lb(judge):
        token = judge.upper().strip('\n').split(' ')
```

```
        pa = token[0]
        p = cal2('<s>',pa)
        for i in range(1, len(token)):
                if not token[i] in orderDict[token[i-1]]:
                        print 'no %s followed by %s' % (token[i-1], token[i])
                        break
                else:
                        p *= cal2(token[i-1],token[i])
        return log(p*1.0)

def cal2(val1, val2):
        return orderDict[val1][val2]/tokenDict[val1];

def lm(judge, r):
        token = judge.upper().strip('\n').split(' ')
        pa = token[0]
        p = cal2('<s>',pa)*(1-r)
        for i in range(1,len(token)):
                if not token[i] in orderDict[token[i-1]]:
                        p *= (1-r)*cal1(token[i])
                else:
                        p *= (1-r)*cal1(token[i]) + r*cal2(token[i-1],token[i])
        return log(p*1.0)



voc = open('vocab.txt','r')
tokenDict = {}
tokenList = []
for token in voc.readlines():
        token = token.strip('\n');
        tokenList.append(token)
        tokenDict[token] = 0

uni = open('unigram.txt','r')
totalCount = 0
index = 0
for count in uni.readlines():
        tokenDict[tokenList[index]] = int(count)
        totalCount += int(count)
        index += 1

# (a)
for token in tokenList:
```

```
        if token[0] == 'B':
                print 'for word %s, the probality is %f' % (token, tokenDict[token]*1.0 /
totalCount)


bi = open('bigram.txt', 'r')
orderDict = {}
for line in bi.readlines():
        line = line.split('\t')
        i1 = int(line[0]) - 1
        i2 = int(line[1]) - 1
        count2 = float(line[2])
        if not tokenList[i1] in orderDict.keys():
                orderDict[tokenList[i1]] = {}
        orderDict[tokenList[i1]].update({tokenList[i2]:count2})

#(b)
b = sorted(orderDict['ONE'].items(), key = lambda x: x[1], reverse = True)
for i in range(0,10):
  print ('The word %s has probality %f') % (b[i][0], b[i][1]*1.0/tokenDict['ONE'])

#(c)
print lu('The stock market fell by one hundred points last week')
print lb('The stock market fell by one hundred points last week')
#(d)
print lu('The fourteen officials sold fire insurance')
print lb('The fourteen officials sold fire insurance')

#(e)
pp = []
rr = []
for r in range(0,100,10):
        pp.append(lm('The fourteen officials sold fire insurance',r*1.0/100))
        rr.append(r*1.0/100)

print pp
print rr

plt.plot(rr,pp)
plt.show()
```