```
'''
     Name: Ruiqing Qiu
     PID: A98022702
     Userid: rqiu
     filename: hw3.py
'''
import matplotlib.pyplot as plt
import math
# A list to keep track of all vocabulary
vocab = []
# A list to keep track all the frequency of the vocab in unigram
unigram = []
#Line number 500
lineNumber = 500
# Total number of frequency in unigram
total = 0
AWordsProb = []

#Open vocab.txt and put them into the list
with open ('vocab.txt') as vocabFile:
  for line in vocabFile:
    tmp = line.split()
    vocab.append(tmp[0])
#Open unigrams.txt and put them into the list
with open ('unigram.txt') as unigramFile:
  for line in unigramFile:
    tmp = line.split()
    unigram.append(tmp[0])

print "3.3 Part A, all words start with A and probability: "
index = 1

#Getting the total number
for freq in unigram:
    total += int(freq)
print "Total is %d" % total
for word, freq in zip(vocab,unigram):
    if word[0] == "A":
        print "%d, word is: " % index + word
        print "unigram probability is %.14f\n" % ((float(freq))/total)
        AWordsProb.append(float(freq)/total)
        index += 1

print "\n3.3 Part B, ten most likely words to follow \"THE\"."
#find the word THE in vocab
THEindex = 0
wordsFollowTHE = []
wordsFollowTHEFreq = []
for word in vocab:
    if word == "THE":
        print "\"THE\" is at index %d of the file" % THEindex
        break
    THEindex += 1
print vocab[THEindex]
with open ('bigram.txt') as bigramFile:
  for line in bigramFile:
      tmp = line.split()
      previousWord = tmp[0]
      if previousWord == "4":
        currentWord = tmp[1]
        wordsFollowTHE.append(currentWord)
        frequency = tmp[2]
        wordsFollowTHEFreq.append(frequency)
      if previousWord == "5":
          break
THETotal = 0
for freq in wordsFollowTHEFreq:
    THETotal += int(freq)
print "Count of THE is : %d" % THETotal

TupleList = []
for words,freq in zip(wordsFollowTHE, wordsFollowTHEFreq):
    TupleList.append((int (freq), int(words)))
```

```
#reverse sort the tuple list
TupleList.sort(reverse=True)

#Count to get the top 10 elements
count = 1
for item in TupleList:
    if count == 11:
        break
    else:
        print "%d : the vocabulary is %s" % (count, vocab[item[1]-1])
        print "the bigram probability %.14f\n" % (item[0]/float(THETotal))
        count += 1

print "\n3.3 Part c, comparing Log-likelihood of unigram and bigram"
sentence = "THE STOCK MARKET FELL BY ONE HUNDRED POINTS LAST WEEK"
sentenceWords = sentence.split()

#Calculating unigram log-likelihood
unigramLogProb = 0.0
index = 0
for word in vocab:
  for i in sentenceWords:
      if i == word:
          unigramLogProb += math.log( (float (unigram[index])/total))
  index += 1
print "The unigram Log-likelihood is %.14f" % unigramLogProb

#A function that calculates unigram probability
def unigramProb (curr):
  index = 0
  for word in vocab:
      if curr == word:
          return (float(unigram[index])/total)
      index += 1

#A function that calculates bigram probability based on curr and previous
def bigramProb (curr, previous):
  previousIndex = 1
  wordsFollow = []
  wordsFollowFreq = []
  for word in vocab:
      if word == previous:
        #print "\"%s\"" %previous+" is at index %d of the file"%previousIndex
        break
      previousIndex += 1
  with open('bigram.txt') as bigramFile:
    for line in bigramFile:
        tmp = line.split()
        previousWord = tmp[0]
        if previousWord == str(previousIndex):
            currentWord = tmp[1]
            #print previousWord
            #print currentWord
            wordsFollow.append(currentWord)
            frequency = tmp[2]
            wordsFollowFreq.append(frequency)
        if previousWord == str(previousIndex+1):
            break
  prevTotal = 0
  for freq in wordsFollowFreq:
      prevTotal += int(freq)
  #print "Count of the \"%s\": %d" % (previous,prevTotal)
  index = 0
  currIndex = 0
  for word in vocab:
      if word == curr:
          break;
      currIndex += 1
  for word in wordsFollow:
      if word == str(currIndex+1):
          #print "Bigram Probability: %.14f" % (float(wordsFollowFreq[index])/prevTo
tal)
          return (float(wordsFollowFreq[index])/prevTotal)
        index += 1
    return 0.0

#Calculating bigram log-likelihood
totalBi = 0.0
totalBi += math.log(bigramProb("THE", "<s>"))
for i in range(0, len(sentenceWords)-1):
  totalBi += math.log(bigramProb(sentenceWords[i+1],sentenceWords[i]))
  #print "curr: " + sentenceWords[i+1]
  #print "prev: " + sentenceWords[i] + "\n\n"
print "The bigram Log-likelihood is %.14f" %totalBi
print "\nTherefore, the bigram model yields the highest log-likelihood"
print "\n3.3 Part d, pair that are not observed"
#The sentence
sentence = "THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE"
sentenceWords = sentence.split()

unigramLogProb = 0.0
index = 0
for word in vocab:
    for i in sentenceWords:
        if i == word:
            unigramLogProb += math.log((float(unigram[index])/total))
    index += 1
print "The unigram Log-likelihood is %.14f\n\n" % unigramLogProb

Bi = 0.0
valid = True
Bi += bigramProb("THE", "<s>")
for i in range (0, len(sentenceWords)-1):
    tmp = bigramProb(sentenceWords[i+1],sentenceWords[i])
    if tmp == 0.0:
        print "not observed in training corpus: curr: %s, previous %s" % (sentenceWo
rds[i+1], sentenceWords[i])
        valid = False
        continue
    Bi += math.log(bigramProb(sentenceWords[i+1],sentenceWords[i]))
if valid == True:
  print "The bigram Log-likelihood is %.14f" %Bi
else:
  print "Some Pairs not find in traning corpus, " + "Log(0) undefined, can't calcula
te bigram probability"

print "\n3.3 Part e, mixture model"
Lambda = 0.00
mixModelLogProb = []
mixModelTupleList = []
sentence = "THE SIXTEEN OFFICIALS SOLD FIRE INSURANCE"
sentenceWords = sentence.split()
while Lambda <= 1.0:
  subtotal = 0.0
  subtotal += math.log((1-Lambda) * unigramProb(sentenceWords[0]) + Lambda * bigramP
rob(sentenceWords[0],"<s>"))
  for i in range(0, len(sentenceWords)-1):

    subtotal += math.log((1-Lambda) * unigramProb(sentenceWords[i+1]) + Lambda * big
ramProb(sentenceWords[i+1],sentenceWords[i]))
  mixModelLogProb.append(subtotal)
  mixModelTupleList.append((subtotal,Lambda))
  print "Lambda: %.2f total is %.14f" %(Lambda, subtotal)
  Lambda += 0.01
mixModelTupleList.sort(reverse=True)
print "The best Lambda value is: %.2f" %mixModelTupleList[0][1]
plt.plot(mixModelLogProb)
plt.ylabel('mixture Log-likelihood')
plt.show()
```