**3.5a**

**Solution:**

A 0.018407

AND 0.017863

AT 0.004313

AS 0.003992

AN 0.002999

ARE 0.002990

ABOUT 0.001926

AFTER 0.001347

ALSO 0.001310

ALL 0.001182

A. 0.001026

ANY 0.000632

AMERICAN 0.000612

AGAINST 0.000596

ANOTHER 0.000428

AMONG 0.000374

AGO 0.000357

ACCORDING 0.000348

AIR 0.000311

ADMINISTRATION 0.000292

AGENCY 0.000280

AROUND 0.000277

AGREEMENT 0.000263

AVERAGE 0.000259

ASKED 0.000258

ALREADY 0.000249

AREA 0.000231

ANALYSTS 0.000226

ANNOUNCED 0.000227

ADDED 0.000221

ALTHOUGH 0.000214

AGREED 0.000212

APRIL 0.000207

AWAY 0.000202

**3.5b**

**Solution:**

THE <UNK> 0.028999

THE U. 0.000631

THE FIRST 0.000553

THE COMPANY 0.000550

THE NEW 0.000446

THE UNITED 0.000409

THE GOVERNMENT 0.000321

THE NINETEEN 0.000314

THE SAME 0.000296

THE TWO 0.000290

**3.5c1**

**Solution:**

Likelihood: -64.5094403436

**3.5c2**

**Solution:**

Likelihood: -40.9181321338

Which model yields the highest log-likelihood?

-41 > -64.5 the Bigram model yields the highest log-likelihood

**3.5d1**

**Solution**

Log Likelihood: -44.2919344731

**3.5d2**

**Solution:**

Log Likelihood: -25.3146249355

Which pairs of adjacent words in this sentence are not observed in the training corpus?

<SIXTEEN, OFFICIALS>

<SOLD, FIRE>

What effect does this have on the log-likelihood from the bigram model?

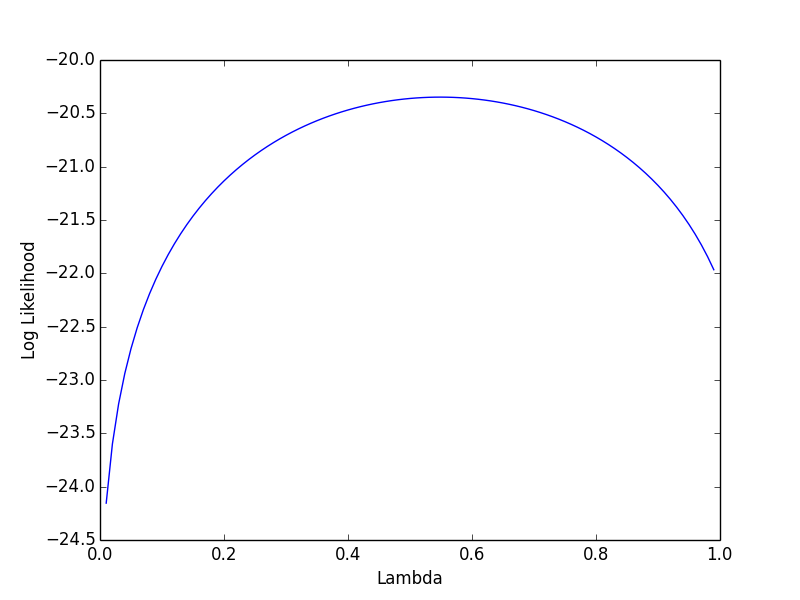
It increases the likelihood, which should not be the case in a real prediction model.

**3.5e**

**Solution:**

Likelihood: -20.3479915601

Lambda: 0.55



**3.5a**

**Source code:**

with open('vocab.txt') as f:

vocab = f.read().splitlines();

with open('unigram.txt') as f:

unigrams = f.read().splitlines();

corpus = []

total = float(0.00)

for freq in unigrams:

total += int(freq)

print freq

print total

for i in range (500):

row = []

corpus.append(row)

row.append(i)

row.append(vocab[i])

row.append(float(unigrams[i])/float(total))

#print corpus

for s in corpus:

if s[1].startswith("A"):

print "%-15s%f" %(s[1], s[2])

**3.5b**

**Source code:**

with open('vocab.txt') as f:

vocab = f.read().splitlines()

with open('unigram.txt') as f:

unigrams = f.read().splitlines()

total = 0

for u in unigrams:

total += int(u)

with open('bigram.txt') as f:

bigrams = []

temp = f.read().splitlines()

for t in temp:

row = t.split("\t")

row[0] = vocab[int(row[0]) - 1]

row[1] = vocab[int(row[1]) - 1]

row[2] = float(row[2])/float(total)

if row[0] == "THE":

bigrams.append(row)

bigrams.sort(key=lambda x: x[2], reverse=True)

for b in range (10):

print "%-5s%-12s%f" %(bigrams[b][0], bigrams[b][1], bigrams[b][2])

**3.5c1**

**Source code:**

import math

sentence = ['THE', 'STOCK', 'MARKET', 'FELL', 'BY', 'ONE', 'HUNDRED', 'POINTS', 'LAST', 'WEEK']

with open('vocab.txt') as f:

vocab = f.read().splitlines()

with open('unigram.txt') as f:

unigrams = f.read().splitlines()

total\_words = 0.00

log = float(0.00)

for u in unigrams:

total\_words += int(u)

print total\_words

for word in sentence:

log += math.log(float(unigrams[vocab.index(word)])/float(total\_words))

print "%-10s%f" %(word, log)

print "Log Likelihood:" + str(log)

#Log Likelihood:-64.5094403436

**3.5c2**

**Source code:**

import math

sentence = ['<s>', 'THE', 'STOCK', 'MARKET', 'FELL', 'BY', 'ONE', 'HUNDRED', 'POINTS', 'LAST', 'WEEK']

with open('vocab.txt') as f:

vocab = f.read().splitlines()

with open('unigram.txt') as f:

unigrams = f.read().splitlines()

with open('bigram.txt') as f:

bigrams = []

temp = f.read().splitlines()

for t in temp:

row = t.split("\t")

row[0] = vocab[int(row[0]) - 1]

row[1] = vocab[int(row[1]) - 1]

row[2] = int(row[2])

bigrams.append(row)

total\_words = 0.00

log = float(0.00)

for i in range (len(sentence)):

# if(i == 0):

# log += math.log(float(unigrams[vocab.index(sentence[i])])/float(unigrams[vocab.index(sentence[i + 1])]))

# else:

for b in bigrams:

if(b[0] == sentence[i-1] and b[1] == sentence[i]):

print b[0] + "\t" + b[1] + "\t" + str(log)

log += math.log(float(b[2])/float(unigrams[vocab.index(sentence[i-1])]))

print "Log Likelihood:" + str(log)

#Log Likelihood:-40.9181321338

**3.5d1**

**Source Code:**

import math

sentence = ['THE', 'SIXTEEN', 'OFFICIALS', 'SOLD', 'FIRE', 'INSURANCE']

with open('vocab.txt') as f:

vocab = f.read().splitlines()

with open('unigram.txt') as f:

unigrams = f.read().splitlines()

total\_words = 0.00

log = float(0.00)

for u in unigrams:

total\_words += int(u)

for word in sentence:

log += math.log(float(unigrams[vocab.index(word)])/float(total\_words))

print word +"\t" + str(log)

print "Log Likelihood:" + str(log)

#Log Likelihood:-44.2919344731

3.5d2

Source Code:

import math

sentence = ['<s>', 'THE', 'SIXTEEN', 'OFFICIALS', 'SOLD', 'FIRE', 'INSURANCE']

with open('vocab.txt') as f:

vocab = f.read().splitlines()

with open('unigram.txt') as f:

unigrams = f.read().splitlines()

with open('bigram.txt') as f:

bigrams = []

temp = f.read().splitlines()

for t in temp:

row = t.split("\t")

row[0] = vocab[int(row[0]) - 1]

row[1] = vocab[int(row[1]) - 1]

row[2] = int(row[2])

bigrams.append(row)

total\_words = 0.00

log = float(0.00)

for i in range (len(sentence)):

# if(i == 0):

# log += math.log(float(unigrams[vocab.index(sentence[i])])/float(unigrams[vocab.index(sentence[i + 1])]))

# else:

found = False

for b in bigrams:

if(b[0] == sentence[i-1] and b[1] == sentence[i]):

found = True

word1 = b[0]

word2 = b[1]

freq = b[2]

#print b[0] + "\t" + b[1] + "\t" +str(found)

elif(b[0] == sentence[i-1] and b[1] == '<UNK>'):

word1\_unk = b[0]

word2\_unk = b[1]

freq\_unk = b[2]

#print b[0] + "\t" + b[1] + "\t" +str(found)

elif(b[0] == '<UNK>' and b[1] == sentence[i]):

unk\_word1 = b[0]

unk\_word2 = b[1]

unk\_freq = b[2]

#print b[0] + "\t" + b[1] + "\t" +str(found)

if(found == True):

print word1 + "\t" + word2 + "\t" + str(log)

log += math.log(float(freq)/float(unigrams[vocab.index(sentence[i-1])]))

else:

print word1\_unk + "\t" + word2\_unk +"\t" + str(log)

#log += math.log(float(freq\_unk)/float(unigrams[vocab.index(sentence[i-1])]))

log += math.log(1)

found = False

print "Log Likelihood:" + str(log)

#Log Likelihood:-30.61667048

**3.5e**

**Source Code:**

import math, matplotlib.pyplot as plt

sentence = ['<s>', 'THE', 'SIXTEEN', 'OFFICIALS', 'SOLD', 'FIRE', 'INSURANCE']

with open('vocab.txt') as f:

vocab = f.read().splitlines()

with open('unigram.txt') as f:

unigrams = f.read().splitlines()

with open('bigram.txt') as f:

bigrams = []

temp = f.read().splitlines()

for t in temp:

row = t.split("\t")

row[0] = vocab[int(row[0]) - 1]

row[1] = vocab[int(row[1]) - 1]

row[2] = int(row[2])

bigrams.append(row)

total\_unigram\_words = 0

for u in unigrams:

total\_unigram\_words += int(u)

total\_words = 0.00

log = []

lamb = []

for n in range (1,100):

lamb.append(float(n)/float(100));

#print lamb

log.append(0);

for i in range (len(sentence)):

# if(i == 0):

# log[n-1] += math.log((1-lamb[n-1])\*(float(unigrams[vocab.index(sentence[i])])/float(unigrams[vocab.index(sentence[i + 1])])) + (lamb[n-1])\*(float(unigrams[vocab.index(sentence[i])])/float(total\_unigram\_words)))

# else:

found = False

for b in bigrams:

if(b[0] == sentence[i-1] and b[1] == sentence[i]):

found = True

word1 = b[0]

word2 = b[1]

freq = b[2]

#print b[0] + "\t" + b[1] + "\t" +str(found)

elif(b[0] == sentence[i-1] and b[1] == '<UNK>'):

word1\_unk = b[0]

word2\_unk = b[1]

freq\_unk = b[2]

#print b[0] + "\t" + b[1] + "\t" +str(found)

elif(b[0] == '<UNK>' and b[1] == sentence[i]):

unk\_word1 = b[0]

unk\_word2 = b[1]

unk\_freq = b[2]

#print b[0] + "\t" + b[1] + "\t" +str(found)

if(found == True):

#print word1 + "\t" + word2 + "\t" + str(log)

log[n-1] += math.log((1-lamb[n-1])\*(float(freq)/ float(unigrams[vocab.index(sentence[i-1])])) + (lamb[n-1])\*(float(unigrams[vocab.index(word1)])/float(total\_unigram\_words)))

else:

#print word1\_unk + "\t" + word2\_unk +"\t" + str(log)

#log[n-1] += math.log((1-lamb[n-1])\*(float(freq\_unk)/float(unigrams[vocab.index(sentence[i-1])])) + (lamb[n-1])\*(float(unigrams[vocab.index(word1\_unk)])/float(total\_unigram\_words)))

log += math.log(1)

found = False

print "Log Likelihood: " + str(log[n-1]) + "\tLambda: " + str(lamb[n-1])

#print log

#Log Likelihood:-28.6677492443

#Max:Log Likelihood: -26.1151701541 Lambda: 0.26

plt.plot(lamb, log)

print max(log)

plt.xlabel('Lambda')

plt.ylabel('Log Likelihood')

plt.show()