In [44]:
```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
import math
```

In [45]:
```
ClassNumber = {'acq': 1,
 'crude': 2,
 'earn': 3,
 'grain': 4,
 'interest': 5,
 'money-fx': 6,
 'ship': 7,
 'trade': 0}
```

In [46]:
```
def addClass(currentClass):
    val = ClassDict.get(ClassNumber[currentClass])
    if(val != None):
        ClassDict[ClassNumber[currentClass]] = val+1
    else:
        tempDict = {ClassNumber[currentClass]:1}
        ClassDict.update(tempDict)
        tempDict = {ClassNumber[currentClass]:0}
        ClassWordCount.update(tempDict)
```

In [47]:
```
def addWordToVocab(word,currentClass):
    val = vocabDict.get(word)
    if(val==None):
        countMat = np.matrix([[0.00],[0.00],[0.00],[0.00],[0.00],[0.00],[0.00],[0
        countMat[ClassNumber[currentClass]] = 1.00
        tempDict = {word:countMat}
        vocabDict.update(tempDict)
        countMat = np.matrix([[0.00],[0.00],[0.00],[0.00],[0.00],[0.00],[0.00],[0
        countMat[ClassNumber[currentClass]] = 1.00
        tempDict = {word:countMat}
        vocabProb.update(tempDict)
    else:
        countMat = val
        countMat[ClassNumber[currentClass]] = countMat[ClassNumber[currentClass]]
        vocabDict[word] = countMat
        countMat = vocabProb.get(word)
        countMat[ClassNumber[currentClass]] = countMat[ClassNumber[currentClass]]
        vocabProb[word] = countMat
    ClassWordCount[ClassNumber[currentClass]] = ClassWordCount[ClassNumber[curren
```

In [48]:
```
Classes = []
```

In [49]:
```
ClassDict = {}
ClassWordCount = {}
```

In [50]:
```
vocabDict = {}
vocabProb = {}
sampleCount = 0
vocabLen = 0
```

In [51]:
```
with open("r8-train-all-terms.txt",'r') as f:
    for line in f:
        c = 0
        sampleCount = sampleCount + 1
        currentClass = ""
        for word in line.lower().split():
            vocabLen = vocabLen+1
            if(c==0):
                currentClass = word
                Classes.append(currentClass)
                addClass(currentClass)
            else:
                addWordToVocab(word,currentClass)
            c=c+1
    f.close()
```

```
C:\Program Files\Anaconda3\lib\site-packages\ipykernel\__main__.py:3: FutureWar
ning: comparison to `None` will result in an elementwise object comparison in t
he future.
  app.launch_new_instance()
```

In [52]:
```
ClassDict
```

Out[52]:
```
{0: 251, 1: 1596, 2: 253, 3: 2840, 4: 41, 5: 190, 6: 206, 7: 108}
```

In [53]:
```
sampleCount
```

Out[53]:
```
5485
```

In [54]:
```
for key in vocabProb.keys():
    val = vocabProb[key]
    val[0] = math.log(val[0] + 1)- math.log(ClassWordCount[0]+vocabLen)
    val[1] = math.log(val[1] + 1)- math.log(ClassWordCount[1]+vocabLen)
    val[2] = math.log(val[2] + 1)- math.log(ClassWordCount[2]+vocabLen)
    val[3] = math.log(val[3] + 1)- math.log(ClassWordCount[3]+vocabLen)
    val[4] = math.log(val[4] + 1)- math.log(ClassWordCount[4]+vocabLen)
    val[5] = math.log(val[5] + 1)- math.log(ClassWordCount[5]+vocabLen)
    val[6] = math.log(val[6] + 1)- math.log(ClassWordCount[6]+vocabLen)
    val[7] = math.log(val[7] + 1)- math.log(ClassWordCount[7]+vocabLen)
```

In [55]:
```
print(vocabProb["extending"])
print(vocabDict["extending"])
```

```
[[-12.27245322]
 [-11.16174191]
 [-12.25886432]
 [-13.56724405]
 [-13.28982845]
 [-13.31274735]
 [-13.33372904]
 [-12.20468502]]
[[  2.]
 [ 10.]
 [  2.]
 [  0.]
 [  0.]
 [  0.]
 [  0.]
 [  2.]]
```

In [56]:
```
prediction = np.matrix([[1.00],[1.00],[1.00],[1.00],[1.00],[1.00],[1.00],[1.00]])
outputClass = []
```

In [57]:
```
math.log(prediction[0,0])
```

Out[57]:
```
0.0
```

In [58]:
```
#Prediction
predictionMatrix = []
outputClass = []
with open("r8-test-all-terms.txt",'r') as f:
    for line in f:
        prediction = np.matrix([[0.00],[0.00],[0.00],[0.00],[0.00],[0.00],[0.00],
        currentClass = ""
        c=0
        for word in line.lower().split():
            if(c==0):
                currentClass = word
                outputClass.append(currentClass)
                #addClass(currentClass)
            else:
                if(vocabProb.get(word) != None):
                    prob = vocabProb[word]
                    prediction[0] = prediction[0]+prob[0]
                    prediction[1] = prediction[1]+prob[1]
                    prediction[2] = prediction[2]+prob[2]
                    prediction[3] = prediction[3]+prob[3]
                    prediction[4] = prediction[4]+prob[4]
                    prediction[5] = prediction[5]+prob[5]
                    prediction[6] = prediction[6]+prob[6]
                    prediction[7] = prediction[7]+prob[7]
            c=c+1
        prediction[0] = prediction[0]+math.log(ClassDict[0]/sampleCount)
        prediction[1] = prediction[1]+math.log(ClassDict[1]/sampleCount)
        prediction[2] = prediction[2]+math.log(ClassDict[2]/sampleCount)
        prediction[3] = prediction[3]+math.log(ClassDict[3]/sampleCount)
        prediction[4] = prediction[4]+math.log(ClassDict[4]/sampleCount)
        prediction[5] = prediction[5]+math.log(ClassDict[5]/sampleCount)
        prediction[6] = prediction[6]+math.log(ClassDict[6]/sampleCount)
        prediction[7] = prediction[7]+math.log(ClassDict[7]/sampleCount)
        predictionMatrix.append(prediction)
    f.close()
```

```
C:\Program Files\Anaconda3\lib\site-packages\ipykernel\__main__.py:15: FutureWa
rning: comparison to `None` will result in an elementwise object comparison in
 the future.
```

In [59]:
```
print(predictionMatrix[0])
```

```
[[-6377.87343512]
 [-6302.96492584]
 [-6909.35259204]
 [-6565.7049638 ]
 [-7901.47015859]
 [-7405.1335676 ]
 [-7014.40410258]
 [-7563.85494205]]
```

In [60]:
```
predictionMatrix[0][0]
```

Out[60]:
```
matrix([[-6377.87343512]])
```

In [61]:
```
Output = []
for i in range(0,len(predictionMatrix)):
    o = 'NA'
    m = max(predictionMatrix[i])
    if(m==predictionMatrix[i][0]):
        o='trade'
    if(m==predictionMatrix[i][1]):
        o='acq'
    if(m==predictionMatrix[i][2]):
        o='crude'
    if(m==predictionMatrix[i][3]):
        o='earn'
    if(m==predictionMatrix[i][4]):
        o='grain'
    if(m==predictionMatrix[i][5]):
        o='interest'
    if(m==predictionMatrix[i][6]):
        o='money-fx'
    if(m==predictionMatrix[i][7]):
        o='ship'
    Output.append(o)
```

In [22]:
```
text_file = open("OutputTest-new.txt", "w")
for k in range(0,len(Output)):
    text_file.write(Output[k]+'\n')
text_file.close()
```

In [62]:
```
ConfusionMatrix = np.zeros(shape=(8,8),dtype=int)
```

In [63]:
```
ConfusionMatrix
```

Out[63]:
```
array([[0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0]])
```

In [64]:
```
ConfusionMatrix[ClassNumber[outputClass[0]],ClassNumber[Output[0]]]
```

Out[64]:
```
0
```

In [65]:
```
for k in range(0,len(Output)):
    val = ConfusionMatrix[ClassNumber[outputClass[k]],ClassNumber[Output[k]]]
    val=val+1
    ConfusionMatrix[ClassNumber[outputClass[k]],ClassNumber[Output[k]]] = val
```

In [66]:
```
print(ConfusionMatrix)
```

```
[[  13   46    0   16    0    0    0    0]
 [   0  694    0    2    0    0    0    0]
 [   0   60   48   13    0    0    0    0]
 [   0   24    0 1059    0    0    0    0]
 [   0    6    0    4    0    0    0    0]
 [   1   58    0   14    0    6    2    0]
 [   1   64    0   12    0    0   10    0]
 [   0   34    1    1    0    0    0    0]]
```

In [102]:
```
correct = 0
wrong = 0
for k in range(0,len(Output)):
    if(Output[k]==outputClass[k]):
        correct = correct + 1
    else:
        wrong = wrong +1
```

In [103]:
```
print(correct)
print(wrong)
```

```
1923
266
```