In [2]:

```
#Imports
import numpy as np
import scipy.special as special
import scipy.optimize
import time

#diGamma func from scipy, use this in your code!
diGamma = special.digamma

#Function definitions for maximizing the VI parameters. This will later be complete
def maxVIParam(phi, gamma, B, alpha, M, k, Wd, eta):

    for d in range(M):
        N = len(Wd[d])
        #Initialization of vars, as shown in E-step.
        phi[d] = np.ones((N,k))*1.0/k
        gamma[d] = np.ones(k)*(N/k) + alpha
        converged = False
        j = 0 #you can use this to print the update error to check your code in the
        '''if(j%10==0 and d==0):
                    print("u e: ", updateError)'''
        #YOUR CODE FOR THE E-STEP HERE
        gamma_old = np.zeros(k)
        while np.sum(abs(np.array(gamma[d])-np.array(gamma_old))) > eta:
            #print("HEY",abs(np.array(gamma[d])-np.array(gamma_old)))
            for n in range(N):
                for i in range(k):
                    phi[d][n,i] = B[i,Wd[d][n]]*np.exp(diGamma(gamma[d][i]))
                phi[d][n,:]=phi[d][n,:]/np.sum(phi[d][n,:])
            gamma_old = gamma[d]
            gamma[d] = alpha + np.sum(phi[d],axis=0)
    return gamma, phi

#Function definitions for maximizing the B parameter. This will later be completed
def MaxB(B, phi, k, V, M, Wd):

    #YOUR CODE FOR THE M-STEP HERE
    for d in range(M):
        for n in range(len(Wd[d])):
            B[:,Wd[d][n]] += phi[d][n,:]

    return B
```

In [24]:

```
#representation of top words for each topic:
nTop = 20
for i in range(k):
    topVocabs = np.argsort(B[i])[-nTop:][::-1]
    topWords = np.array(vectorizer.get_feature_names())[topVocabs]
    print("top words for topic ",i,": ")
    print(topWords)
```

```
top words for topic  0 :
['people' 'god' 'father' 'spirit' 'question' 'religion' 'know' 'say'
'son'
 'church' 'exist' 'world' 'like' 'group' 'existence' 'need' 'creed' 'h
oly'
 'use' 'post']
top words for topic  1 :
['space' 'nasa' 'year' 'launch' 'satellite' 'use' 'gov' 'mission'
 'project' 'post' 'data' 'orbit' 'access' 'earth' 'program' 'research'
 'probe' 'nntp' 'high' 'design']
top words for topic  2 :
['team' 'flyer' 'play' 'game' 'hockey' 'ca' 'gm' 'year' 'player' 'seas
on'
 'point' 'goal' 'leaf' 'city' 'good' 'win' 'time' 'nntp' 'think' 'pos
t']
top words for topic  3 :
['car' 'say' 'god' 'jesus' 'know' 'think' 'hell' 'believe' 'christian'
 'people' 'time' 'thing' 'die' 'come' 'really' 'make' 'good' 'like' 'w
ay'
 'truth']
```

For our training the following are the topics:
0)**Space**
1)**Cars**
2)**Christian Religion**
3)**Hockey**

Since there are no guarantees regarding the order of the topics (LDA is unsupervised) or what your initial B matrix values were, it is difficult to say exactly what results you should be seeing. Hopefully, you can see the four original topics to some extent in your result. For example, one of my topics had top words like "christ", and "god", meaning it was most likely the topic for "Christian Religion" documents, while another literally had the words "Hockey" and "NHL" in it. Our vocabulary is as mentioned earlier quite limited, so it may be possible that one of your topics is a bit unclear or close to another. You can also load the α and β values in the cell below which are pre-calculated for 200 iterations and compare your topic results to those available there.

In [34]:

```
eta = 10e-5 #threshold for convergence

#we are not re-initializing beta and alpha, we calculated them using the training d

V = len(vectorizer.get_feature_names()) #vocab. cardinality
M = int(len(testDocs)) #number of documents
k = 4 #amount of emotions

#variational params (one for each doc)
phi = [None]*M
gamma = [None]*M
WdTest = [None]*M

'''Same magic from before to get the word matrix correct, replace this if you redid


'''Now that you have your variables initialized for the test documents, you should
maximizing the VI parameters with those variables instead. Remember, we're just cal
gamma and phi for each test document so there is no iteration between maximizing Be

for d in range(M):
    Wmat = vectorizer.transform([testDocs[d]]).toarray()[0] #get vocabulary matrix
    WVidxs = np.where(Wmat!=0)[0]
    WVcounts = Wmat[WVidxs]
    N = np.sum(WVcounts)
    W = np.zeros((N)).astype(int)

    i = 0
    for WVidx, WV in enumerate(WVidxs):
        for wordCount in range(WVcounts[WVidx]):
            W[i] = WV
            i+=1
    WdTest[d] = W #We save the list of words for the document for analysis later

# YOUR CODE to Run the gamma/phi maximization here.
    N = len(W)
    #Initialization of vars, as shown in E-step.
    phi[d] = np.ones((N,k))*1.0/k
    gamma[d] = np.ones(k)*(N/k) + alpha
    converged = False
    j = 0 #you can use this to print the update error to check your code in the beg
    '''if(j%10==0 and d==0):
                print("u e: ", updateError)'''
    #YOUR CODE FOR THE E-STEP HERE
    gamma_old = np.zeros(k)
    while np.amax(abs(np.array(gamma[d])-np.array(gamma_old))) > eta:
        #print("HEY",abs(np.array(gamma[d])-np.array(gamma_old)))
        for n in range(N):
            for i in range(k):
                phi[d][n,i] = B[i,W[n]]*np.exp(diGamma(gamma[d][i]))
            phi[d][n,:]=phi[d][n,:]/np.sum(phi[d][n,:])
        gamma_old = gamma[d]
        gamma[d] = alpha + np.sum(phi[d],axis=0)
```

We have now calculated the variational parameters for our test documents, so let us see what information we can infer from them. If you take a look at the pseudo code we used for the MaxVIParam method, you can see

In [18]:

```
#representation of top words for each topic:
## 4 topics
### RESULTS:
####  0 economy
####  1 bush
####  2 israel
####  3 soviet

nTop = 20
for i in range(k):
    topVocabs = np.argsort(B[i])[-nTop:][::-1]
    topWords = np.array(vectorizer.get_feature_names())[topVocabs]
    print("top words for topic ",i,": ")
    print(topWords)
```

```
top words for topic  0 :
['percent' 'year' 'new' 'prices' 'million' 'company' 'billion' 'oil'
 'rose' 'price' 'rate' 'bank' 'market' 'month' 'gold' 'york' 'dollar'
 'report' 'thursday' 'fell']
top words for topic  1 :
['people' 'year' 'police' 'bush' 'new' '000' 'city' 'state' 'day' 'yea
rs'
 'man' 'officials' 'reported' 'dukakis' 'mrs' 'national' 'family' 'lik
e'
 'today' 'county']
top words for topic  2 :
['president' 'israel' 'new' 'trade' 'shamir' 'union' 'officials' 'unit
ed'
 'million' 'american' 'foreign' 'states' 'state' 'group' 'news' 'minis
ter'
 'jewish' 'administration' 'time' 'japan']
top words for topic  3 :
['soviet' 'government' 'gorbachev' 'party' 'president' 'people' 'unio
n'
 'court' 'house' 'year' '000' 'defense' 'national' 'barry' 'committee'
 'military' 'communist' 'new' 'years' 'fbi']
```

In [14]:

```
#representation of top words for each topic:
## 4 topics
### RESULTS:
####  0 happy music
####  1 rock and party
####  2 indie
####  3 melancholic

nTop = 20
for i in range(k):
    topVocabs = np.argsort(B[i])[-nTop:][::-1]
    topWords = np.array(vectorizer.get_feature_names())[topVocabs]
    print("top words for topic ",i,": ")
    print(topWords)
```

```
top words for topic  0 :
['away' 'feel' 'god' 'need' 'pain' 'lord' 'free' 'joy' 'hey' 'heart'
 'happy' 'sun' 'soul' 'shot' 'makin' 'make' 'goes' 'run' 'kiss' 'dar
k']
top words for topic  1 :
['war' 'rock' 'fame' 'need' 'mind' 'shy' 'start' 'ooh' 'let' 'feel' 'e
ye'
 'hey' 'try' 'gonna' 'say' 'white' 'wonder' 'er' 'party' 'night']
top words for topic  2 :
['home' 'right' 'burn' 'wanna' 'say' 'think' 'let' 'good' 'life' 'toni
ght'
 'night' 'people' 'hate' 'bed' 'second' 'heart' 'day' 'turn' 'seen'
 'sweet']
top words for topic  3 :
['girl' 'lonely' 'good' 'ooh' 'easy' 'man' 'tell' 'walking' 'say' 'lit
tle'
 'angel' 'ain' 'fi' 'honey' 'change' 'huh' 'gun' 'need' 'smell' 'lon
g']
```

In [ ]:

```

```