In [91]:
```
import nltk
```

In [92]:
```
# Question 2: Tweets with Positive sentiments

pos_tweets = [('I love this car', 'positive'),
              ('This view is amazing', 'positive'),
              ('I feel great this morning', 'positive'),
              ('I am so excited about the concert', 'positive'),
              ('He is my best friend', 'positive')]
```

In [93]:
```
# Question 2: Tweets with negative sentiments

neg_tweets = [('I do not like this car', 'negative'),
             ('This view is horrible', 'negative'),
             ('I feel tired this morning', 'negative'),
             ('I am not looking forward to the concert', 'negative'),
             ('He is my enemy', 'negative')]
```

In [94]:
```
# Question 3: Creating a single list with first element as array containing th
e words and second element is the sentiment

tweets = []
```

In [95]:
```
# Question 3: Creating a single list with first element as array containing th
e words and second element is the sentiment
# Also getting rid of words smaller than 2 characters and converting all text
 to lowercase

for (words, sentiment) in pos_tweets + neg_tweets:
    array = [w.lower() for w in words.split() if len(w) >= 3]
    tweets.append((array, sentiment))
```

In [96]:
```
# Printing the tweets tuple

tweets
```

Out[96]:
```
[(['love', 'this', 'car'], 'positive'),
 (['this', 'view', 'amazing'], 'positive'),
 (['feel', 'great', 'this', 'morning'], 'positive'),
 (['excited', 'about', 'the', 'concert'], 'positive'),
 (['best', 'friend'], 'positive'),
 (['not', 'like', 'this', 'car'], 'negative'),
 (['this', 'view', 'horrible'], 'negative'),
 (['feel', 'tired', 'this', 'morning'], 'negative'),
 (['not', 'looking', 'forward', 'the', 'concert'], 'negative'),
 (['enemy'], 'negative')]
```

In [98]:

```
# Question 4: List of test tweets

test_tweets = [
        (['feel', 'happy', 'this', 'morning'], 'positive'),
        (['larry', 'friend'], 'positive'),
        (['not', 'like', 'that', 'man'], 'negative'),
        (['house', 'not', 'great'], 'negative'),
        (['your', 'song', 'annoying'], 'negative')]
```

In [102]:

```
# Question 5: Importing nltk.proability package with FreqDist class

from nltk.probability import FreqDist
from nltk.tokenize import word_tokenize
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\jadonvs\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[102]:

```
True
```

In [100]:

```
# Question 5: Collecting all the words from the tweets

freq_tweets = "love this car this view amazing feel great this morning excited about the concert best friend not like this car this view horrible feel tired this morning not looking forward the concert enemy"
```

In [103]:

```
# Question 5: Function to produce a frequency distribution that encodes how often each word occurs in a text

fdist = FreqDist(word.lower() for word in word_tokenize(freq_tweets))
```

In [104]:

```
# Question 5: List of distinct words ordered by frequency of appearance

fdist
```

Out[104]:

```
FreqDist({'about': 1,
          'amazing': 1,
          'best': 1,
          'car': 2,
          'concert': 2,
          'enemy': 1,
          'excited': 1,
          'feel': 2,
          'forward': 1,
          'friend': 1,
          'great': 1,
          'horrible': 1,
          'like': 1,
          'looking': 1,
          'love': 1,
          'morning': 2,
          'not': 2,
          'the': 2,
          'this': 6,
          'tired': 1,
          'view': 2})
```

In [114]:

```
# Question 6: Use the Naive Bayes classifier to train a classifier

from nltk.classify.naivebayes import NaiveBayesClassifier
```

In [124]:

```
# Question 6: The list of word features need to be extracted from the tweets.
# We use the following function to get the list plus the two helper functions.

def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
      all_words.extend(words)
    return all_words

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features

word_features = get_word_features(get_words_in_tweets(tweets))
```

In [126]:

```
# Question 6: Printing word_features

word_features
```

Out[126]:

```
dict_keys(['love', 'this', 'car', 'view', 'amazing', 'feel', 'great', 'mornin
g', 'excited', 'about', 'the', 'concert', 'best', 'friend', 'not', 'like', 'h
orrible', 'tired', 'looking', 'forward', 'enemy'])
```

In [149]:

```
# Question 6: To create a classifier, we need to decide what features are re
levant. To do that, we first need a feature extractor.
# The one we are going to use returns a dictionary indicating what words are
 contained in the input passed.
# Here, the input is the tweet. We use the word features list defined above
 along with the input to create the dictionary.

def extract_features(document):
   document_words = set(document)
   features = {}
   for word in word_features:
        features['contains(%s)' % word] = (word in document_words)
   return features
```

In [150]:

```
# Question 6:
# With our feature extractor, we can apply the features to our classifier usin
g the method apply_features.
# We pass the feature extractor along with the tweets list defined above.

training_set = nltk.classify.apply_features(extract_features, tweets)
```

In [151]:

```
# Printing Training Set

training_set
```

Out[151]:

```
[({'contains(love)': True, 'contains(this)': True, 'contains(car)': True, 'co
ntains(view)': False, 'contains(amazing)': False, 'contains(feel)': False, 'c
ontains(great)': False, 'contains(morning)': False, 'contains(excited)': Fals
e, 'contains(about)': False, 'contains(the)': False, 'contains(concert)': Fal
se, 'contains(best)': False, 'contains(friend)': False, 'contains(not)': Fals
e, 'contains(like)': False, 'contains(horrible)': False, 'contains(tired)': F
alse, 'contains(looking)': False, 'contains(forward)': False, 'contains(enem
y)': False}, 'positive'), ({'contains(love)': False, 'contains(this)': True,
'contains(car)': False, 'contains(view)': True, 'contains(amazing)': True, 'c
ontains(feel)': False, 'contains(great)': False, 'contains(morning)': False,
'contains(excited)': False, 'contains(about)': False, 'contains(the)': False,
'contains(concert)': False, 'contains(best)': False, 'contains(friend)': Fals
e, 'contains(not)': False, 'contains(like)': False, 'contains(horrible)': Fal
se, 'contains(tired)': False, 'contains(looking)': False, 'contains(forwar
d)': False, 'contains(enemy)': False}, 'positive'), ...]
```

In [152]:

```
# Now that we have our training set, we can train our classifier.

classifier = nltk.NaiveBayesClassifier.train(training_set)
```

In [185]:

```
# Question 7:
# Calculate the classification accuracy of the trained classifier using the
 training set.

from nltk import classify
from nltk import NaiveBayesClassifier

classifier = nltk.NaiveBayesClassifier.train(training_set)
accuracy = classify.accuracy(classifier, training_set)
print(accuracy)

print (classifier.show_most_informative_features(10))
```

```
1.0
Most Informative Features
           contains(not) = False          positi : negati =      1.6 : 1.0
          contains(love) = False          negati : positi =      1.2 : 1.0
         contains(great) = False          negati : positi =      1.2 : 1.0
       contains(excited) = False          negati : positi =      1.2 : 1.0
      contains(horrible) = False          positi : negati =      1.2 : 1.0
         contains(tired) = False          positi : negati =      1.2 : 1.0
       contains(amazing) = False          negati : positi =      1.2 : 1.0
         contains(about) = False          negati : positi =      1.2 : 1.0
        contains(friend) = False          negati : positi =      1.2 : 1.0
       contains(forward) = False          positi : negati =      1.2 : 1.0
None
```