In [1]:
```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn import cross_validation
from sklearn.cross_validation import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

#import matplotlib.pylab as plt
#%matplotlib inline
#plt.rcParams['figure.figsize'] = 10, 8

np.random.seed(36)
```

Creates the train feature file

In [2]:
```
with open("data/train.txt",'r') as f1:
    with open("features_BoW_train.csv", "w") as f2:
        f2.truncate()
        f2.write("Text")
        f2.write(",")
        f2.write("Topic")
        f2.write("\n")
        for line in f1:
            topic, text = line.split("\t")
            f2.write(text.replace("\n",""))
            f2.write(",")
            f2.write(topic)
            f2.write("\n")
```

Creates the text feature file

In [3]:
```
with open("data/test.txt",'r') as f1:
    with open("features_BoW_test.csv", "w") as f2:
        f2.truncate()
        f2.write("Text")
        f2.write(",")
        f2.write("Topic")
        f2.write("\n")
        for line in f1:
            topic, text = line.split("\t")
            f2.write(text.replace("\n",""))
            f2.write(",")
            f2.write(topic)
            f2.write("\n")
```

In [4]:
```
traindata = pd.read_csv("features_BoW_train.csv")
testdata = pd.read_csv("features_BoW_test.csv")
```

In [5]:
```
testdata.head(5)
```

Out[5]:

	Text	Topic
0	asian exporters fear damage japan rift mountin...	trade
1	china daily vermin eat pct grain stocks survey...	grain
2	australian foreign ship ban ends nsw ports hit...	ship
3	sumitomo bank aims quick recovery merger sumit...	acq
4	amatil proposes two for bonus share issue amat...	earn

In [6]:
```
X_train = traindata['Text']
Y_train = traindata['Topic']
```

In [7]:
```
X_test = testdata['Text']
Y_test = testdata['Topic']
```

In [8]:
```
# Fit a counter
tfidf_vectorizer = TfidfVectorizer(sublinear_tf=True, max_df=0.5,
                                 stop_words='english')
tfidf_vectorizer.fit(X_train)

# Transform to a counter
X_train_tfidf = tfidf_vectorizer.transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

In [9]:
```
model = MultinomialNB(alpha=0.01)
model.fit(X_train_tfidf, Y_train)
```

Out[9]:
```
MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)
```

In [10]:
```
pred = model.predict(X_test_tfidf)
print "F1 Accuracy = %.4f" %metrics.f1_score(Y_test, pred, average='weighted')
print "Accuracy = %.4f" %metrics.accuracy_score(Y_test, pred)
print "Classification Report: \n" + metrics.classification_report(Y_test, pred)
```

```
F1 Accuracy = 0.9458
Accuracy = 0.9461
Classification Report: 
             precision    recall  f1-score   support

        acq       0.95      0.95      0.95       696
      crude       0.91      0.97      0.94       121
       earn       0.97      0.97      0.97      1083
      grain       1.00      0.90      0.95        10
   interest       0.92      0.75      0.83        81
   money-fx       0.84      0.87      0.85        87
       ship       0.92      0.64      0.75        36
      trade       0.76      0.97      0.85        75

avg / total       0.95      0.95      0.95      2189
```

In [ ]:
```
lr_model = LogisticRegression(multi_class='multinomial',solver='newton-cg')
lr_model.fit(X_train_tfidf, Y_train)
```

In [ ]:
```
pred = lr_model.predict(X_test_tfidf)
print "F1 Accuracy = %.4f" %metrics.f1_score(Y_test, pred, average='weighted')
print "Accuracy = %.4f" %metrics.accuracy_score(Y_test, pred)
print "Classification Report: \n" + metrics.classification_report(Y_test, pred)
```

In [ ]:
```
knn_model = KNeighborsClassifier()
knn_model.fit(X_train_tfidf, Y_train)
```

In [ ]:
```
pred = knn_model.predict(X_test_tfidf)
print "F1 Accuracy = %.4f" %metrics.f1_score(Y_test, pred, average='weighted')
print "Accuracy = %.4f" %metrics.accuracy_score(Y_test, pred)
print "Classification Report: \n" + metrics.classification_report(Y_test, pred)
```

In [ ]:
```
lda_model = LDA()
lda_model.fit(X_train_tfidf, Y_train)
```

In [ ]: