The Reuters dataset, a set of short newswires and their topics, published by Reuters in 1986. It's a simple, wi
toy dataset for text classification. There are 46 different topics; some topics are more represented than other
topic has at least 10 examples in the training set. The set contain 8982 training samples and 2246 test samp
IMDB the Reuters dataset comes packaged as part of Keras.

```
import numpy as np
from keras.datasets import reuters
```

```
vocabulary=7500 # we will use on the first 7500 most used words
```

```
np.load.__defaults__=(None, True, True, 'ASCII')
(train_data, train_labels), (test_data, test_labels) = reuters.load_data(num_words=voc
np.load.__defaults__=(None, False, True, 'ASCII')
```

```
len(train_data)
```

```
8982
```

```
len(test_data)
```

In the next line of code we will print the lists that contain sequences of words represented by a word index.

```
print(train_data[1]) # train_data is a list of word sequences
```

Now we will vectorize the training and test data. Basically we will create a matrix where the rows are the revie
where the columns represent the vocabulary (7500 columns). We will set a 1 in the correct column if the word
review matches a word of the vocabulary. As we limit the reviews to 150 words there will be 7350 places whe
have a zero. This means that matrix will be rather sparse.

```
def vectorize_sequences(sequences, dimension=vocabulary):
    results=np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence]=1
    return results
```

```
x_train=vectorize_sequences(train_data)
x_test=vectorize_sequences(test_data)
```

To vectorize the labels we will use one-hot encoding. One-hot encoding is a widely used format for categorica
also. In this case, one-hot encoding of the labels consists of embedding each label as an all-zero vector with
place of the label index.

```
from keras.utils.np_utils import to_categorical
one_hot_train_labels = to_categorical(train_labels)
one_hot_test_labels = to_categorical(test_labels)
```

```
print(one_hot_train_labels[0])
```

Now we are ready to apply Logostic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(multi_class='multinomial', solver='newton-cg')
model.fit(x_train, train_labels)
score = model.score(x_test, test_labels)
print("Accuracy:", score)
```