```python
# -*- coding: utf-8 -*-
"""
Çok sınıflı sınıflandırma  (Multiclass classification)
* Haber metinleri 46 farklı kategoriye ayrılmıştır
* imdb örneğine benzer şekilde kelimeler 10000 ile sınırlandırılmıştır
* vectorize_sequences fonksiyonuna alternatif olarak Tokenizer kutuphanesinden
   sequences_to_matrix fonksiyonu kullanılmıştır
* Ağ 46 çıkışlı olarak tanımlanmış ve softmax fonksiyonu kullanılmıştır.
* train_label veya test_label 0-45 arası tamsayı değerler sakladığı için
   keras.utils.to_categorical fonksiyonu ile herbir etiket 46 elemanlı
   çıkış vektörüne dönüştürülmüştür. Yani haber 10 numaralı kategorideyse
   vektörde bu konum 1 diğerleri 0 olarak tanımlanır.
"""

import keras
from keras.datasets import reuters
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.preprocessing.text import Tokenizer

#(x_train, y_train), (x_test, y_test) =\
# reuters.load_data(num_words=None, test_split=0.2)
max_words = 10000
(train_data, train_labels), (test_data, test_labels) =\
reuters.load_data(num_words=max_words)


word_index = reuters.get_word_index()

print('# of Training Samples: {}'.format(len(train_data)))
print('# of Test Samples: {}'.format(len(test_data)))

word_index = reuters.get_word_index()


num_classes = max(train_labels) + 1

print('# of Classes: {}'.format(num_classes))

index_to_word = {}
for key, value in word_index.items():
    index_to_word[value] = key

print(' '.join([index_to_word[x] for x in train_data[0]]))
print(train_labels[0])


#10000 kelime için 10000 elemanlı binary vektör oluştur
#ilgili yorum içerisinde kullanılan kelimeler 1 diğerleri 0 olarak işaretlenir
tokenizer = Tokenizer(num_words=max_words)
x_train = tokenizer.sequences_to_matrix(train_data, mode='binary')
x_test = tokenizer.sequences_to_matrix(test_data, mode='binary')

#one-hot coding
#46 elemanlı (her sınıf için) binary vektör
y_train = keras.utils.to_categorical(train_labels, num_classes)
y_test= keras.utils.to_categorical(test_labels, num_classes)

#model
model = Sequential()
model.add(Dense(32,activation='relu', input_shape=(10000,)))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dense(46,activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])

print(model.metrics_names)
batch_size =32
epochs = 10
```

```
73
74  history = model.fit(x_train,
75                      y_train,
76                      batch_size=batch_size,
77                      epochs=epochs, #verbose=1,
78                      validation_split=0.1)#!! #veya validation_data=(x_val, y_val)),
79
80
81  #Eğitilen modelin test verleri için başarımını belirle
82  score = model.evaluate(x_test,
83                         y_test,
84                         batch_size=batch_size)
85
86
87  print('Test loss:', score[0])
88  print('Test accuracy:', score[1])
89
90  # Grafik çizim işlemleri
91  import matplotlib.pyplot as plt
92
93  loss = history.history['loss']
94  val_loss = history.history['val_loss']
95
96  epochs = range(1, len(loss) + 1)
97
98  plt.figure(1)
99  plt.plot(epochs, loss, 'bo', label='Training loss')
100 plt.plot(epochs, val_loss, 'b', label='Validation loss')
101 plt.title('Training and validation loss')
102 plt.xlabel('Epochs')
103 plt.ylabel('Loss')
104 plt.legend()
105 plt.show()
106
107 plt.figure(2)
108 acc = history.history['acc']
109 val_acc = history.history['val_acc']
110 acc = history.history['acc']
111 val_acc = history.history['val_acc']
112 plt.plot(epochs, acc, 'ro', label='Training acc')
113 plt.plot(epochs, val_acc, 'r', label='Validation acc')
114 plt.title('Training and validation accuracy')
115 plt.xlabel('Epochs')
116 plt.ylabel('acc')
117 plt.legend()
118
119 plt.show()
120
```