지영

LSTM으로 로이터 뉴스 카테고리 분석

```python
from keras.datasets import reuters
from keras.models import Sequential
from keras.layers import Dense, LSTM, Embedding
from keras.preprocessing import sequence
from keras.utils import to_categorical

import numpy
import tensorflow as tf
import matplotlib.pyplot as plt

# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.random.set_seed(3)

# 불러온 데이터를 학습셋과 테스트셋으로 나누기
(X_train, Y_train), (X_test, Y_test) = reuters.load_data(num_words=1000, test_split=0.2)

# 데이터 확인하기
category = numpy.max(Y_train) + 1
print(category, '카테고리')
print(len(X_train), '학습용 뉴스기사')
print(len(X_test), '테스트용 뉴스기사')
print(X_train[0])

# 데이터 전처리
x_train = sequence.pad_sequences(X_train, maxlen=100)
x_test = sequence.pad_sequences(X_test, maxlen=100)
y_train = to_categorical(Y_train, category)
y_test = to_categorical(Y_test, category)
```

```
46 카테고리
8982 학습용 뉴스기사
2246 테스트용 뉴스기사
[1, 2, 2, 8, 43, 10, 447, 5, 25, 207…]
```

로이터 뉴스 데이터셋 호출

각 기사의 단어 수가 제각각 다르므로 이를 동일하게 맞추기 위해 데이터 전처리 함수 sequence를 이용

케라스(Keras)의 버전에 따라서 keras.utils 모듈 내에서 np_utils를 불러올 수 없다하여 np_utils 대신에 keras.utils 모듈에서 to_categorical 함수를 직접 사용

데이터 학습셋, 테스트셋으로 나누기, reuter.load_data() 함수로 기사 호출, num_words 매개변수는 빈도가 높은 상위 1000개의 단어만 사용하도록 설정, test_split 인자를 통해 20%를 테스트셋으로 사용

시퀀스 데이터를 패딩 후 모든 시퀀스를 동일한 길이로 만들고, 레이블을 원-핫 인코딩하여 모델에 적용

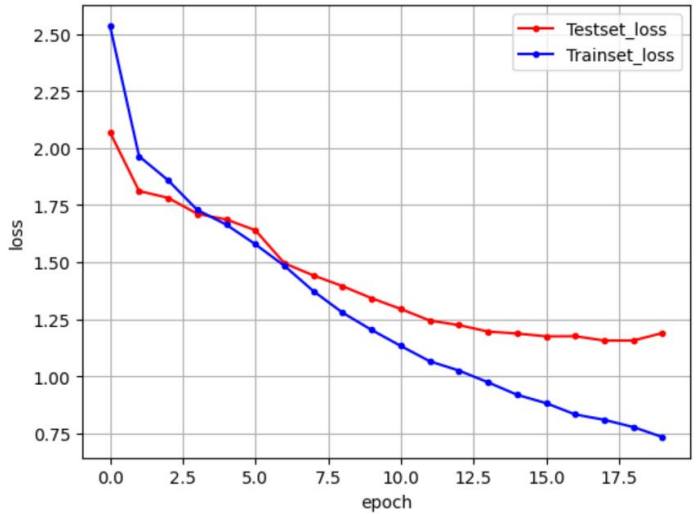
```python
# 모델의 설정
model = Sequential()
model.add(Embedding(1000, 100))
model.add(LSTM(100, activation='tanh'))
model.add(Dense(46, activation='softmax'))

# 모델의 컴파일
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# 모델의 실행
history = model.fit(x_train, y_train, batch_size=100, epochs=20,
                    validation_data=(x_test, y_test))

# 테스트 정확도 출력
print("₩nTest Accuracy: %.4f" % (model.evaluate(x_test, y_test)[1]))

# 테스트셋의 오차
y_vloss = history.history['val_loss']
# 학습셋의 오차
y_loss = history.history['loss']

# 그래프로 표현
x_len = numpy.arange(len(y_loss))
plt.plot(x_len, y_vloss, marker='.', c='red', label='Testset_loss')
plt.plot(x_len, y_loss, marker='.', c='blue', label='Trainset_loss')

plt.legend(loc='upper right')
plt.grid()
plt.xlabel('epoch')
plt.ylabel('loss')
plt.show()
```

Embedding 층으로 단어를 임베딩하고, LSTM 층으로 시퀀스 정보를 학습하며, Dense 층으로 최종 출력을 생성

모델을 컴파일하고 손실 함수, 옵티마이저, 평가 메트릭을 설정

테스트 정확도를 출력하고, 학습 및 테스트셋의 손실을 그래프로 시각화

```
Test Accuracy: 0.7191
```



```
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding, LSTM, Conv1D, MaxPooling1D
from keras.datasets import imdb
import numpy
import tensorflow as tf
import matplotlib.pyplot as plt

# seed 값 설정
seed = 0
numpy.random.seed(seed)
tf.random.set_seed(3)

# 학습셋과 테스트셋 지정하기
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=500)

# 데이터 전처리
x_train = sequence.pad_sequences(x_train, maxlen=100)
x_test = sequence.pad_sequences(x_test, maxlen=100)

# 모델의 설정
model = Sequential()
model.add(Embedding(5000, 100))
model.add(Dropout(0.5))
model.add(Conv1D(64, 5, padding='valid', activation='relu', strides=1))
model.add(MaxPooling1D(pool_size=4))
model.add(LSTM(55))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.summary( )
```

LSTM&CNN 을조합해 영화 리뷰 분류

IMDb 데이터셋을 불러와서 학습셋과 테스트셋으로 나누기,num_words=500는 가장 빈번하게 등장하는 500개의 단어만 사용하도록 설정

클래스가 긍정 또는 부정 두 가지뿐이라 원-핫 인코딩 과정이 없음

Embedding(5000, 100): 5000개의 단어에 대한 100차원의 임베딩 수행

Dropout(0.5): 과적합을 방지

Conv1D, MaxPooling1D: 지역적 특징 추출

LSTM(55): LSTM 층을 추가하여 시퀀스 정보를 학습

```python
# 모델 컴파일
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

# 모델의 실행
history = model.fit(x_train, y_train, batch_size=100, epochs=5, validation_data=(x_test, y_test))

# 테스트 정확도 출력
print("₩nTest Accuracy: %.4f" % (model.evaluate(x_test, y_test)[1]))

# 테스트셋의 오차
y_vloss = history.history['val_loss']
# 학습셋의 오차
y_loss = history.history['loss']

# 그래프로 표현
x_len = numpy.arange(len(y_loss))
plt.plot(x_len, y_vloss, marker='.', c='red', label='Testset Loss')
plt.plot(x_len, y_loss, marker='.', c='blue', label='Trainset Loss')

# 그래프에 그리드를 주고 레이블을 표시
plt.legend(loc='upper right')
plt.grid()
plt.xlabel('epoch')
plt.ylabel('loss')
plt.show()
```

이진 분류 문제이므로
binary_crossentropy 손실 함수와 Adam 사용

테스트 정확도를 출력하고, 학습 및 테스트셋의 손실을 그래프로 시각화