



강사 윤영주
yj.youn1103@gmail.com

다섯째 마당

딥러닝 활용하기

18장 시퀀스 배열로 다루는 순환 신경망(RNN)

- 1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기
- 2 LSTM과 CNN의 조합을 이용한 영화 리뷰
분류하기
- 3 어텐션을 사용한 신경망

시퀀스 배열로 다루는 순환 신경망(RNN)

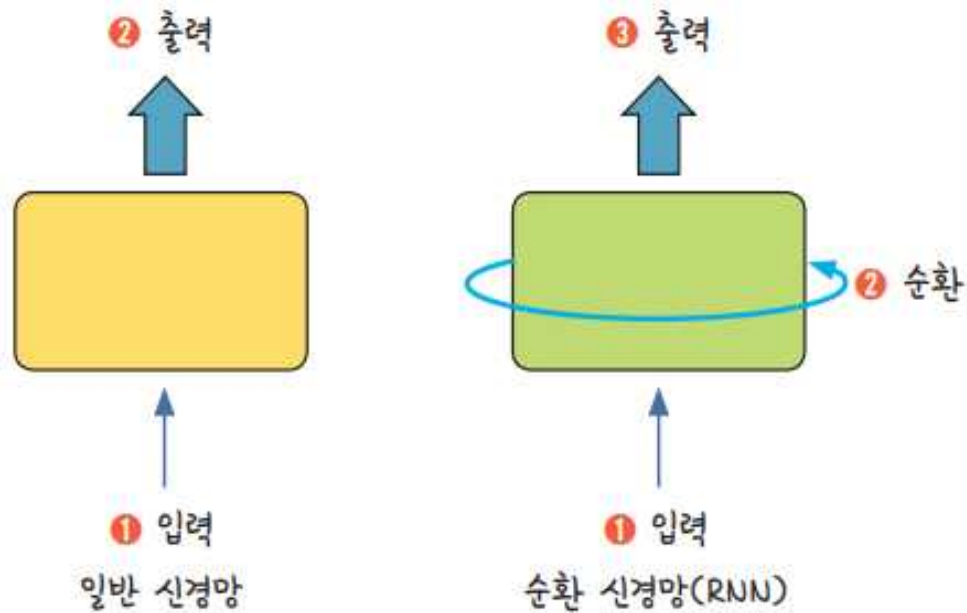
- 시퀀스 배열로 다루는 순환 신경망(RNN)
 - 인공지능이 문장을 듣고 이해한다는 것은 많은 문장을 '이미 학습(train)해 놓았다'는 것
 - 문장을 학습하는 것은 우리가 지금까지 공부한 내용과는 성질이 조금 다름
 - 문장은 여러 개의 단어로 이루어져 있는데, 그 의미를 전달하려면 각 단어가 정해진 순서대로 입력되어야 하기 때문임
 - 즉, 여러 데이터가 순서와 관계없이 입력되던 것과는 다르게, 이번에는 과거에 입력된 데이터와 나중에 입력된 데이터 사이의 관계를 고려해야 하는 문제가 생기는 것

시퀀스 배열로 다루는 순환 신경망(RNN)

- 시퀀스 배열로 다루는 순환 신경망(RNN)
 - 이를 해결하기 위해 순환 신경망(Recurrent Neural Network, RNN) 방법이 고안되었음
 - 순환 신경망은 여러 개의 데이터가 순서대로 입력되었을 때 앞서 입력받은 데이터를 잠시 기억해 놓는 방법
 - 기억된 데이터가 얼마나 중요한지 판단하고 별도의 가중치를 주어 다음 데이터로 넘어감
 - 모든 입력 값에 이 작업을 순서대로 실행하므로 다음 층으로 넘어가기 전에 같은 층을 맴도는 것처럼 보임
 - 이렇게 같은 층 안에서 맴도는 성질 때문에 순환 신경망(이하 RNN)이라고 함

시퀀스 배열로 다루는 순환 신경망(RNN)

▼ 그림 18-1 | 일반 신경망과 순환 신경망(RNN)의 차이





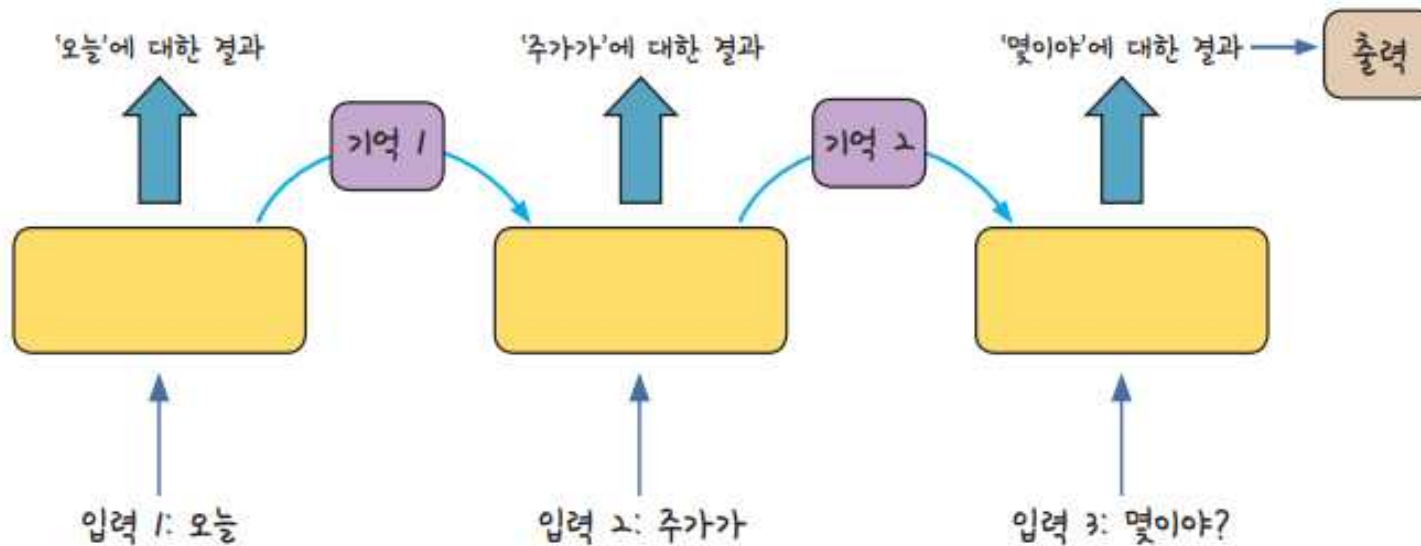
시퀀스 배열로 다루는 순환 신경망(RNN)

- 시퀀스 배열로 다루는 순환 신경망(RNN)
 - 예를 들어 인공지능 비서에게 “오늘 주가가 몇이야?”라고 묻는다고 가정하자
 - 그림 18-1의 '순환 신경망(RNN²)'에 해당하는 '순환' 부분에서는 단어를 하나 처리할 때마다 기억해 다음 입력 값의 출력을 결정
 - 이를 그림으로 표현하면 그림 18-2와 같음



시퀀스 배열로 다루는 순환 신경망(RNN)

▼ 그림 18-2 | “오늘 주가가 몇이야?”를 RNN이 처리하는 방식



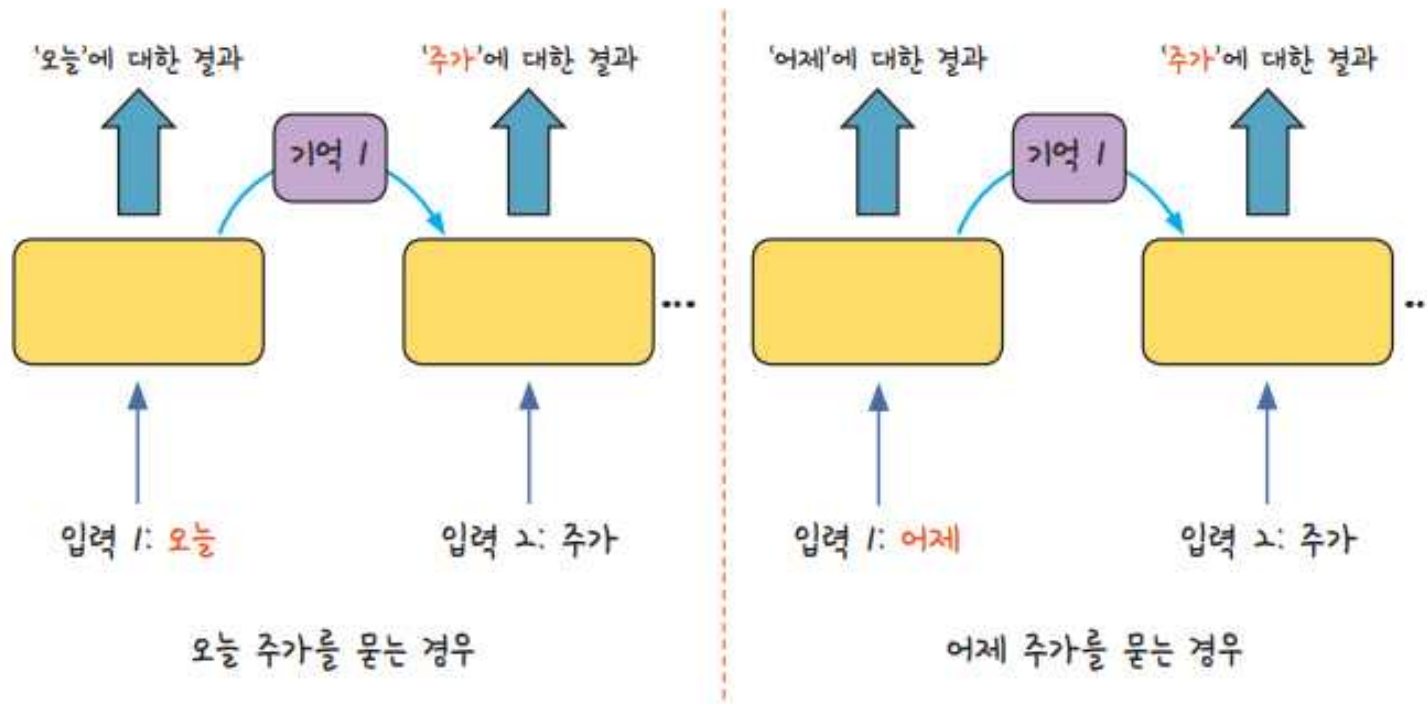
시퀀스 배열로 다루는 순환 신경망(RNN)

- 시퀀스 배열로 다루는 순환 신경망(RNN)
 - 순환되는 중에 앞서 나온 입력에 대한 결과가 뒤에 나오는 입력 값에 영향을 주는 것을 알 수 있음
 - 이렇게 해야 비슷한 두 문장이 입력되었을 때 그 차이를 구별해 출력 값에 반영할 수 있음
 - 예를 들어 그림 18-3에서 입력 2의 값은 양쪽 모두 '주가'이지만, 왼쪽은 '오늘'을 기준으로, 오른쪽은 '어제'를 기준으로 계산되어야 함



시퀀스 배열로 다루는 순환 신경망(RNN)

▼ 그림 18-3 | RNN을 사용하는 이유

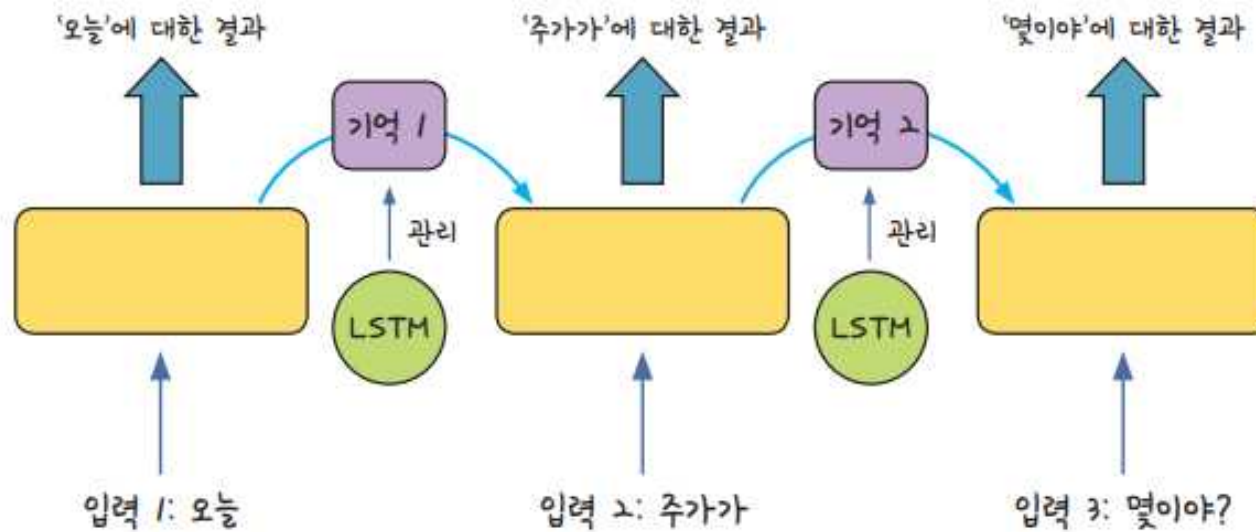


시퀀스 배열로 다루는 순환 신경망(RNN)

- 시퀀스 배열로 다루는 순환 신경망(RNN)
 - RNN이 처음 개발된 이후, RNN의 결과를 더욱 개선하기 위한 노력이 계속되어 왔음
 - 이 중에서 LSTM(Long Short Term Memory) 방법을 함께 사용하는 기법이 현재 가장 널리 사용되고 있음
 - LSTM은 한 층 안에서 반복을 많이 해야 하는 RNN의 특성상 일반 신경망보다 기울기 소실 문제(9.2절. 활성화 함수와 고급 경사 하강법)가 더 많이 발생하고 이를 해결하기 어렵다는 단점을 보완한 방법
 - 즉, 반복되기 직전에 다음 층으로 기억된 값을 넘길지 여부를 관리하는 단계를 하나 더 추가하는 것

시퀀스 배열로 다루는 순환 신경망(RNN)

▼ 그림 18-4 | LSTM은 기억 값의 가중치를 관리하는 장치



시퀀스 배열로 다루는 순환 신경망(RNN)

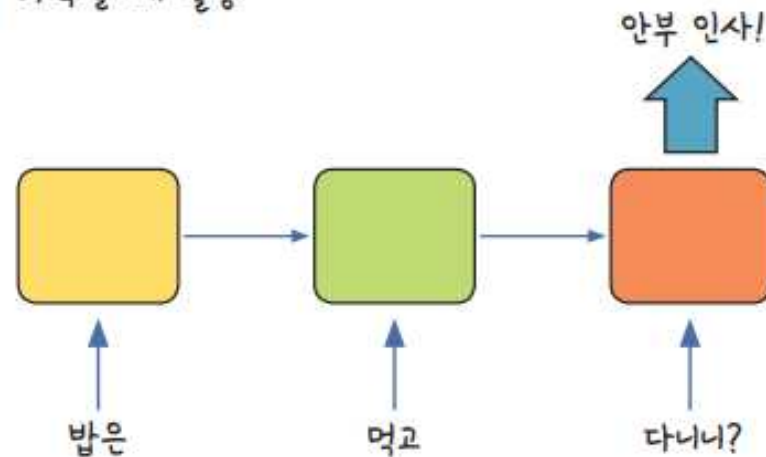
● 시퀀스 배열로 다루는 순환 신경망(RNN)

- RNN 방식의 장점은 입력 값과 출력 값을 어떻게 설정하느냐에 따라 그림 18-5와 같이 여러 가지 상황에서 이를 적용할 수 있다는 것

▼ 그림 18-5 | RNN 방식의 다양한 활용

① 다수 입력 단일 출력

예 문장을 읽고 뜻을 파악할 때 활용

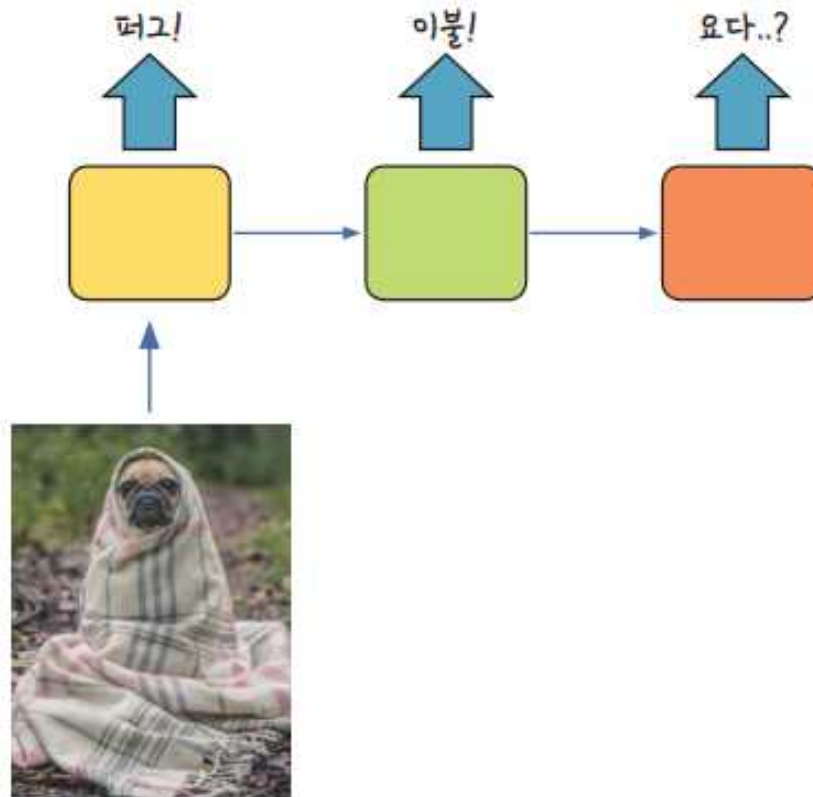




시퀀스 배열로 다루는 순환 신경망(RNN)

② 단일 입력 다수 출력

예 사진의 캡션을 만들 때 활용

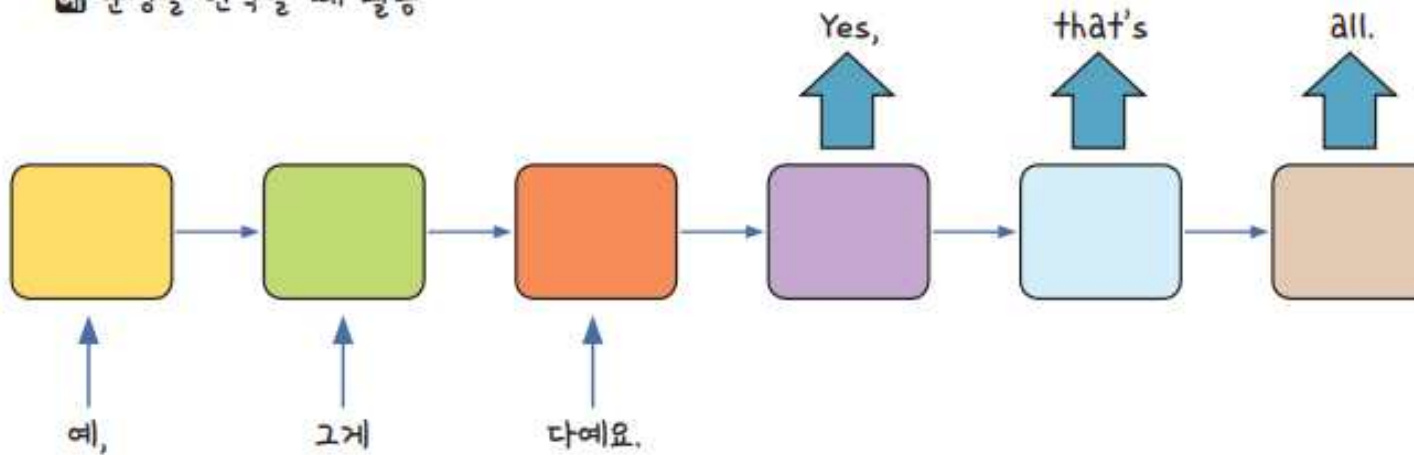




시퀀스 배열로 다루는 순환 신경망(RNN)

④ 다수 입력 다수 출력

예 문장을 번역할 때 활용



시퀀스 배열로 다루는 순환 신경망(RNN)

- 시퀀스 배열로 다루는 순환 신경망(RNN)
 - 케라스는 딥러닝 학습에 필요한 데이터를 쉽게 내려받을 수 있게 `load_data()` 함수를 제공
 - 앞서 살펴본 MNIST 데이터셋 외에도 RNN 학습에 적절한 텍스트 대용량 데이터를 제공
 - 케라스가 제공하는 '로이터 뉴스 카테고리 분류'와 'IMDB 영화 리뷰'를 통해 지금부터 RNN을 학습해 보자



1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기

모두의
딥러닝
계정 3월



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

- 입력된 문장 의미를 파악하는 것은 곧 모든 단어를 종합해 하나의 카테고리로 분류하는 작업이라고 할 수 있음
- 예를 들어 "안녕, 오늘 날씨가 참 좋네."라는 말은 '인사' 카테고리에 분류해야 함
- 다음과 같이 조금 더 길고 전문적인 말도 정확하게 분류

중부 지방은 대체로 맑겠으나, 남부 지방은 구름이 많겠습니다.

→ 날씨

올 초부터 유동성의 힘으로 주가가 일정하게 상승했습니다.

→ 주식

이번 선거에서는 누가 이길 것 같아?

→ 정치

퍼셉트론의 한계를 극복한 신경망이 다시 뜨고 있대.

→ 딥러닝

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 이번에 실습할 내용은 이처럼 긴 텍스트를 읽고 이 데이터가 어떤 의미를 지니는지 카테고리로 분류하는 연습
 - 실습을 위해 로이터 뉴스 데이터를 사용
 - 로이터 뉴스 데이터는 총 1만 1,228개의 뉴스 기사가 46개의 카테고리로 나뉘어진 대용량 텍스트 데이터
 - 데이터는 케라스를 통해 다음과 같이 불러오겠음

```
# 로이터 뉴스 데이터셋 불러오기  
from tensorflow.keras.datasets import reuters
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기

모두의
딥러닝
계정 3월



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 다음과 같이 불러온 데이터를 학습셋과 테스트셋으로 나누겠음

```
# 불러온 데이터를 학습셋과 데이터셋으로 나누기
(X_train, y_train), (X_test, y_test) = reuters.load_data(num_words=1000,
test_split=0.2)
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

- `reuters.load_data()` 함수를 이용해 기사를 불러왔음
- `test_split` 옵션을 통해 20%만 테스트셋으로 사용하겠다고 지정
- 여기서 `num_words` 옵션은 무엇을 의미하는지 알아보고자 먼저 불러온 데이터에 대해 몇 가지를 출력해 보자

```
# 데이터를 확인한 후 출력해 보겠습니다.  
category = np.max(y_train) + 1  
print(category, '카테고리')  
print(len(X_train), '학습용 뉴스 기사')  
print(len(X_test), '테스트용 뉴스 기사')  
print(X_train[0])
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기

모두의
딥러닝
계정 3월



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 출력 결과는 다음과 같음

실행 결과

46 카테고리

8982 학습용 뉴스 기사

2246 테스트용 뉴스 기사

[1, 2, 2, 8, 43, 10, 447, 5, 25, 207...]

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

- 먼저 `np.max()` 함수로 `y_train`의 종류를 구하니 46개의 카테고리로 구분되어 있음을 알 수 있었음(0부터 세기 때문에 1을 더해서 출력)
- 이 중 8,982개는 학습용으로, 2,246개는 테스트용으로 준비되어 있음
- `print(X_train[0])`으로 기사를 출력해 보니 단어가 나오지 않고 [1, 2, 2, 8, 43...] 같은 숫자가 나옴
- 이처럼 딥러닝은 단어를 그대로 사용하지 않고 숫자로 변환한 후 학습할 수 있음
- 여기서는 데이터 안에서 해당 단어가 몇 번이나 나타나는지 세어 빈도에 따라 번호를 붙였음
- 예를 들어 3이라고 하면 세 번째로 빈도가 높은 단어라는 의미
- 이러한 작업을 위해 `tokenizer()` 같은 함수를 사용하는데, 케라스는 이 작업을 이미 마친 데이터를 불러올 수 있음

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 기사 안의 단어 중에는 거의 사용되지 않는 것들도 있음
 - 모든 단어를 다 사용하는 것은 비효율적이므로 빈도가 높은 단어만 불러와 사용
 - 이때 사용하는 인자가 바로 테스트셋과 학습셋으로 나눌 때 함께 적용했던 num_words=1000의 의미
 - 빈도가 1~1,000에 해당하는 단어만 선택해서 불러오는 것

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 또 하나 주의해야 할 점은 각 기사의 단어 수가 제각각 다르므로 이를 동일하게 맞추어야 한다는 것
 - 이때는 다음과 같이 데이터 전처리 함수 sequence()를 이용

```
from tensorflow.keras.preprocessing import sequence

# 단어의 수를 맞추어 줍니다.
X_train = sequence.pad_sequences(X_train, maxlen=100)
X_test = sequence.pad_sequences(X_test, maxlen=100)
```


1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기

모두의
딥러닝
계정 3월



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 여기서 maxlen=100은 단어 수를 100개로 맞추라는 의미
 - 만일 입력된 기사의 단어 수가 100보다 크면 100번째 단어만 선택하고 나머지는 버림
 - 100에서 모자랄 때는 모자라는 부분을 모두 0으로 채움

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 이제 y 데이터에 원-핫 인코딩 처리를 하여 데이터 전처리 과정을 마칩

```
# 원-핫 인코딩 처리를 합니다.  
y_train = to_categorical(y_train)  
y_test = to_categorical(y_test)
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 데이터 전처리 과정이 끝났으므로 딥러닝의 구조를 만들 차례

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

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - Embedding 층과 LSTM 층을 새로 추가
 - Embedding 층은 데이터 전처리 과정을 통해 입력된 값을 받아 다음 층이 알 수 있는 형태로 변환하는 역할
 - Embedding('불러온 단어의 총수', '기사당 단어 수') 형식으로 사용하며, 모델 설정 부분의 맨 처음에 있어야 함
 - LSTM은 앞서 설명했듯이 RNN에서 기억 값에 대한 가중치를 제어하며, LSTM(기사당 단어 수, 기타 옵션) 형식으로 적용
 - LSTM의 활성화 함수로는 tanh를 주로 사용하므로 activation='tanh'로 지정

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

- 이제 다음과 같이 ❶ 모델 실행의 옵션을 정하고 ❷ 조기 중단 설정과 함께 학습을 실행 ❸

모델의 실행 옵션을 정합니다.

```
model.compile(loss='categorical_crossentropy', optimizer='adam',  
metrics=['accuracy']) ..... ❶
```

학습의 조기 중단을 설정합니다.

```
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=5) ..... ❷
```

모델을 실행합니다.

```
history = model.fit(X_train, y_train, batch_size=20, epochs=200, validation_  
data=(X_test, y_test), callbacks=[early_stopping_callback]) ..... ❸
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 앞서 MNIST에 사용되었던 그래프 코드 출력을 더한 전체 코드는 다음과 같음

실습 | LSTM을 이용해 로이터 뉴스 카테고리 분석하기



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

import numpy as np
import matplotlib.pyplot as plt
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

```
# 데이터를 불러와 학습셋, 테스트셋으로 나눕니다.
(X_train, y_train), (X_test, y_test) = reuters.load_data(num_words=1000,
test_split=0.2)

# 데이터를 확인해 보겠습니다.
category = np.max(y_train) + 1
print(category, '카테고리')
print(len(X_train), '학습용 뉴스 기사')
print(len(X_test), '테스트용 뉴스 기사')
print(X_train[0])

# 단어의 수를 맞추어 줍니다.
X_train = sequence.pad_sequences(X_train, maxlen=100)
X_test = sequence.pad_sequences(X_test, maxlen=100)
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기

모두의
딥러닝
계정 3월



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기

```
# 원-핫 인코딩 처리를 합니다.  
y_train = to_categorical(y_train)  
y_test = to_categorical(y_test)  
  
# 모델의 구조를 설정합니다.  
model = Sequential()  
model.add(Embedding(1000, 100))  
model.add(LSTM(100, activation='tanh'))  
model.add(Dense(46, activation='softmax'))
```


1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기

```
# 모델의 실행 옵션을 정합니다.
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# 학습의 조기 중단을 설정합니다.
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=5)

# 모델을 실행합니다.
history = model.fit(X_train, y_train, batch_size=20, epochs=200,
validation_data=(X_test, y_test), callbacks=[early_stopping_callback])

# 테스트 정확도를 출력합니다.
print("\n Test Accuracy: %.4f" % (model.evaluate(x_test, y_test)[1]))
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

```
# 학습셋과 테스트셋의 오차를 저장합니다.  
y_vloss = history.history['val_loss']  
y_loss = history.history['loss']  
  
# 그래프로 표현해 보겠습니다.  
x_len = np.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 표시하겠습니다.  
plt.legend(loc='upper right')  
plt.grid()  
plt.xlabel('epoch')  
plt.ylabel('loss')  
plt.show()
```

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기

실행 결과

Epoch 1/200

450/450 [=====] - 8s 11ms/step - loss: 2.2100 -
accuracy: 0.4390 - val_loss: 1.9456 - val_accuracy: 0.5116

... (중략) ...

Epoch 16/200

450/450 [=====] - 5s 11ms/step - loss: 0.5121 -
accuracy: 0.8691 - val_loss: 1.2316 - val_accuracy: 0.7177

71/71 [=====] - 0s 4ms/step - loss: 1.2316 -
accuracy: 0.7177

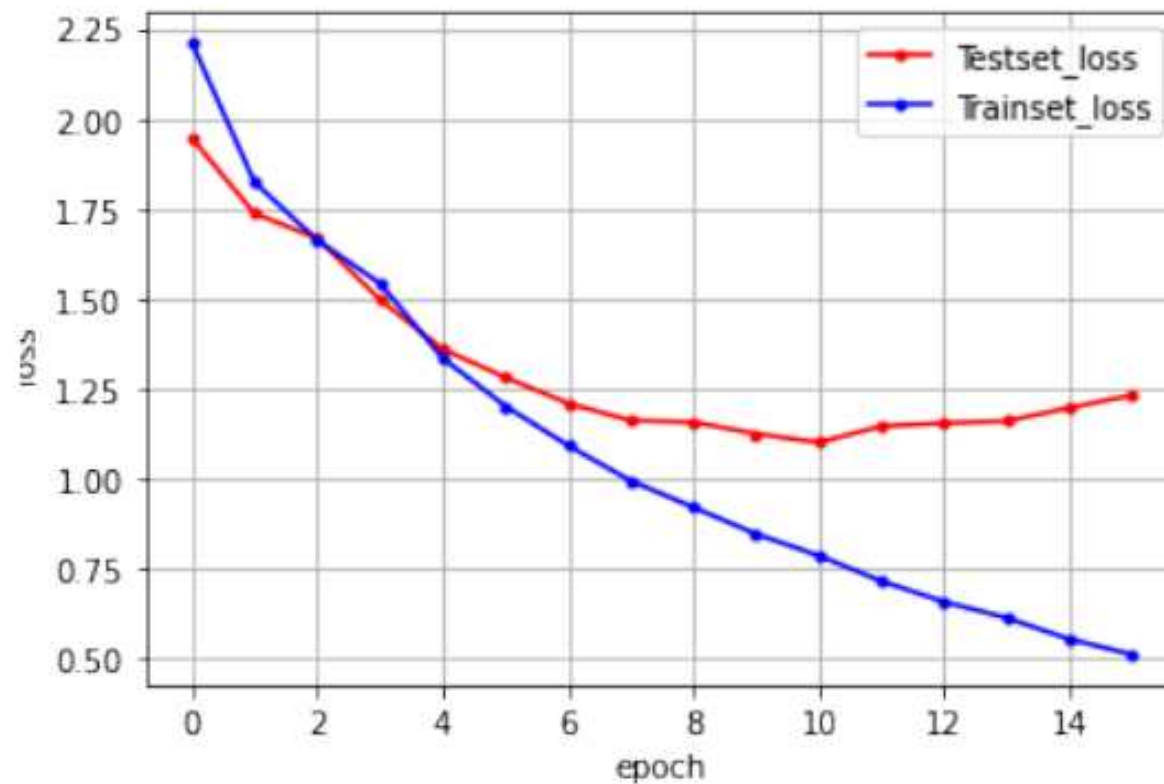
Test Accuracy: 0.7177

1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



● LSTM을 이용한 로이터 뉴스 카테고리 분류하기

▼ 그림 18-6 | 그래프로 확인하는 학습 결과



1 LSTM을 이용한 로이터 뉴스 카테고리 분류하기



- LSTM을 이용한 로이터 뉴스 카테고리 분류하기
 - 16번째 에포크에서 학습이 자동 중단되었으며, 71.7%의 정확도를 보였음
 - 그래프를 통해 테스트셋의 오차가 상승할 때 학습이 멈추었음을 알 수 있음



2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



● LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

- 이번에 사용할 인터넷 영화 데이터베이스(Internet Movie DataBase, IMDB)는 영화와 관련된 정보와 출연진 정보, 개봉 정보, 영화 후기, 평점까지 매우 폭넓은 데이터가 저장된 자료
- 영화에 관해 남긴 2만 5,000여 개의 영화 리뷰가 담겨 있으며, 해당 영화를 긍정적으로 평가했는지 혹은 부정적으로 평가했는지도 담겨 있음
- 앞서 다루었던 로이터 뉴스 데이터와 마찬가지로 각 단어에 대한 전처리를 마친 상태
- 데이터셋에서 나타나는 빈도에 따라 번호가 정해지므로 빈도가 높은 데이터를 불러와 학습시킬 수 있음

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 데이터 전처리 과정은 로이터 뉴스 데이터와 거의 같음
 - 다만 클래스가 긍정 또는 부정 두 가지뿐이라 원-핫 인코딩 과정이 없음

```
# 테스트셋을 지정합니다.  
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=5000)  
  
X_train = sequence.pad_sequences(x_train, maxlen=500)  
X_test = sequence.pad_sequences(x_test, maxlen=500)
```


2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 이제 모델을 다음과 같이 설정
 - model.summary() 함수를 이용해 현재 설정된 모델의 구조를 살펴보자

```
# 모델의 구조를 설정합니다.  
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()
```

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 실행 결과는 다음과 같음

실행 결과

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, None, 100)	500000

dropout_1 (Dropout)	(None, None, 100)	0

conv1d_1 (Conv1D)	(None, None, 64)	32064

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

max_pooling1d_1 (MaxPooling1d)	(None, None, 64)	0
lstm_1 (LSTM)	(None, 55)	26400
dense_1 (Dense)	(None, 1)	56
activation_1 (Activation)	(None, 1)	0

=====

Total params: 558,520

Trainable params: 558,520

Non-trainable params: 0

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



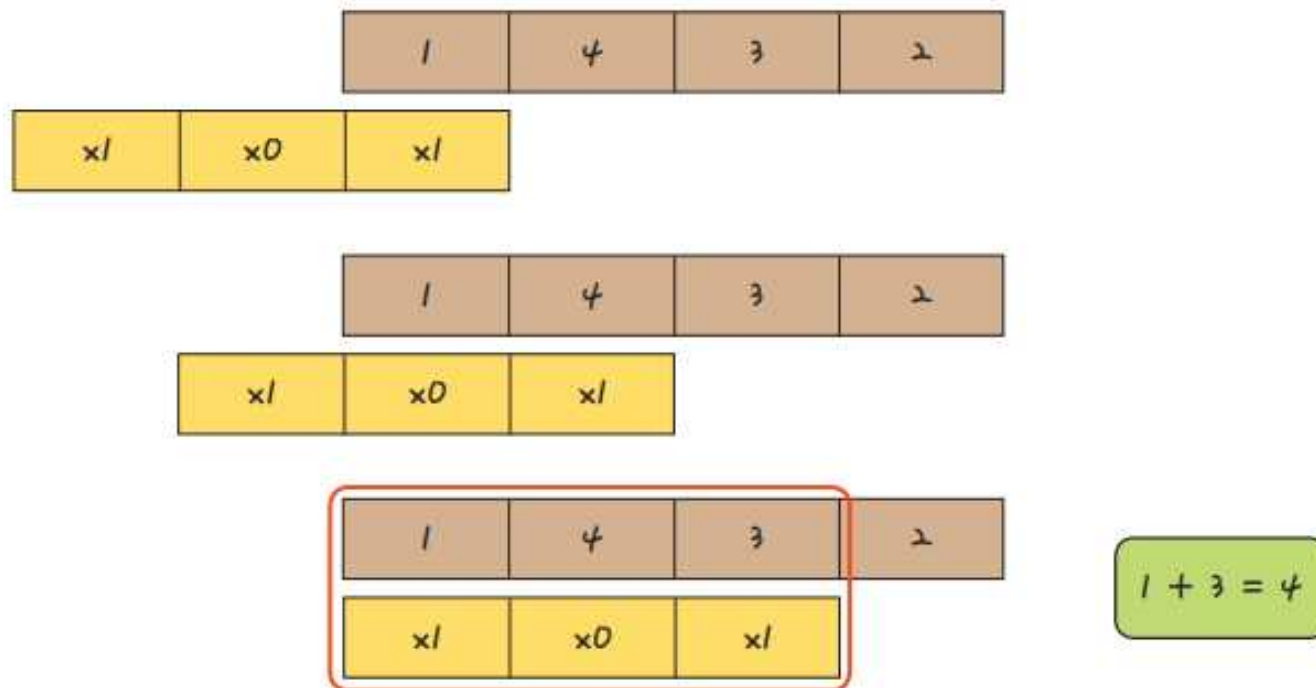
- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 출력 결과에서 우리가 아직 보지 못한 부분은 Conv1D와 MaxPooling1D
 - 2차원 배열을 가진 이미지와는 다르게 지금 다루고 있는 데이터는 배열 형태로 이루어진 1차원이라는 차이가 있음
 - Conv1D는 Conv2D의 개념을 1차원으로 옮긴 것
 - 컨볼루션 층이 1차원이고 이동하는 배열도 1차원

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

모두의
딥러닝
계정 3월

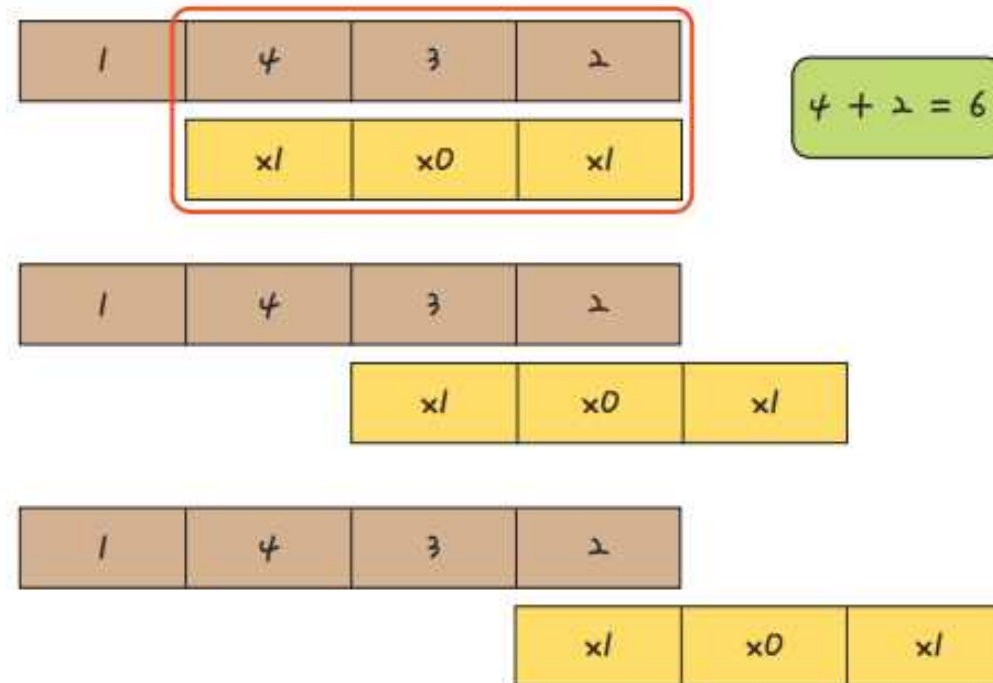


▼ 그림 18-7 | Conv1D의 개념



2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

모두의
딥러닝
계정 3월



2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

모두의
딥러닝
계정 3월



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 노란색으로 표시된 배열이 커널을 의미
 - 이 커널이 지나가면서 원래의 1차원 배열에 가중치를 각각 곱해 새로운 층인 컨볼루션 층을 만들

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - MaxPooling1D 역시 마찬가지로
 - 2차원 배열을 1차원으로 바꾸어 정해진 구역 안에서 가장 큰 값을 다음 층으로 넘기고 나머지는 버림

▼ 그림 18-8 | MaxPooling1D의 개념



2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 전체 코드는 다음과 같음

실습 | LSTM과 CNN을 조합해 영화 리뷰 분류하기



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Embedding,
LSTM, Conv1D, MaxPooling1D
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.callbacks import EarlyStopping

import numpy as np
import matplotlib.pyplot as plt
```

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

```
# 데이터를 불러와 학습셋, 테스트셋으로 나눕니다.
```

```
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=5000)
```

```
# 단어의 수를 맞춥니다.
```

```
X_train = sequence.pad_sequences(X_train, maxlen=500)
```

```
X_test = sequence.pad_sequences(X_test, maxlen=500)
```

```
# 모델의 구조를 설정합니다.
```

```
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))
```

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

```
model.add(LSTM(55))
model.add(Dense(1))
model.add(Activation('sigmoid'))

# 모델의 실행 옵션을 정합니다.
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
# 학습의 조기 중단을 설정합니다.
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=3)

# 모델을 실행합니다.
history = model.fit(X_train, y_train, batch_size=40, epochs=100,
validation_split=0.25, callbacks=[early_stopping_callback])
```

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

```
# 테스트 정확도를 출력합니다.  
print("\n Test Accuracy: %.4f" % (model.evaluate(X_test, y_test)[1]))  
  
# 학습셋과 테스트셋의 오차를 저장합니다.  
y_vloss = history.history['val_loss']  
y_loss = history.history['loss']  
  
# 그래프로 표현해 보겠습니다.  
x_len = np.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')
```

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

모두의
딥러닝
제3화



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

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

2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

실행 결과

```
Epoch 1/100
469/469 [=====] - 18s 17ms/step - loss: 0.4083 -
accuracy: 0.7973 - val_loss: 0.2848 - val_accuracy: 0.8818
... (중략) ...
Epoch 5/100
469/469 [=====] - 7s 16ms/step - loss: 0.1197 -
accuracy: 0.9581 - val_loss: 0.3106 - val_accuracy: 0.8896
782/782 [=====] - 4s 6ms/step - loss: 0.3367 - ac
curacy: 0.8796

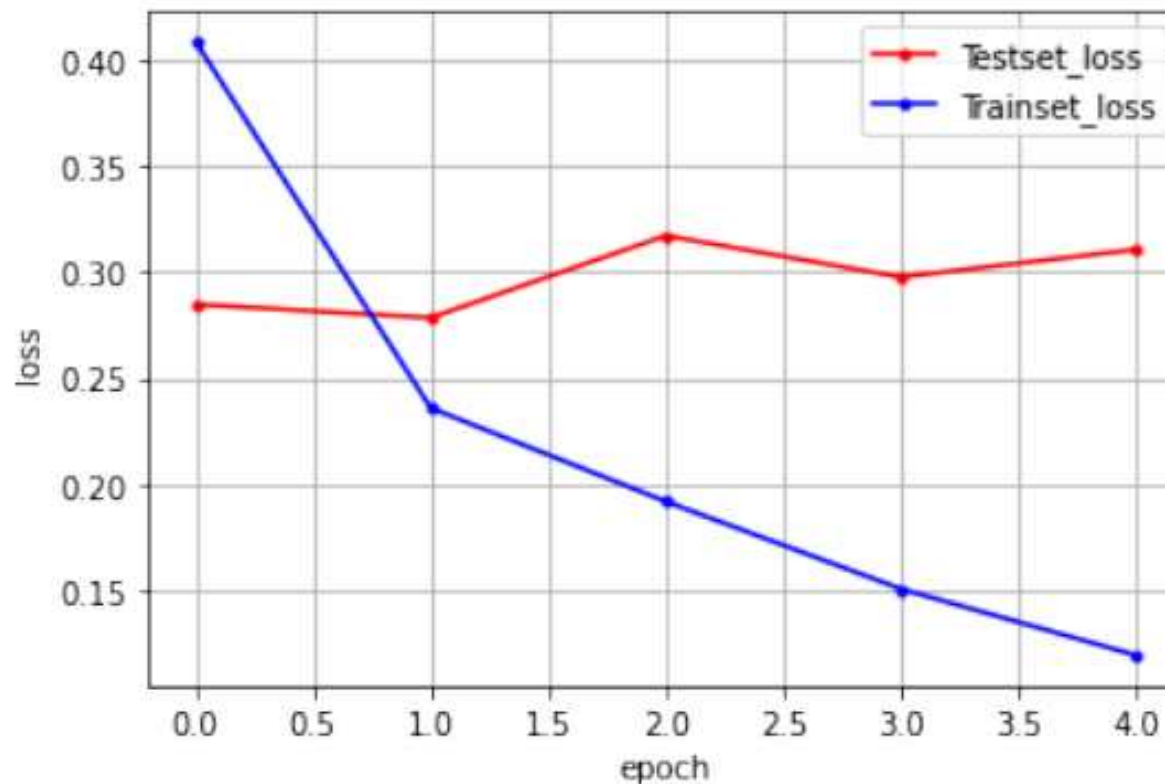
Test Accuracy: 0.8796
```


2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기

▼ 그림 18-9 | 그래프로 확인하는 학습 결과



2 LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기



- LSTM과 CNN의 조합을 이용한 영화 리뷰 분류하기
 - 5번째 에포크에서 학습이 자동 중단되었고 그때의 테스트셋 정확도는 87.96%
 - 그래프를 통해 학습 과정을 확인할 수 있음
 - 학습이 중단되는 시점은 실행마다 달라질 수 있음