

08. 시계열 데이터와 순환 신경망

Preview

■ 시계열 데이터

- 시간 정보가 들어 있는 데이터
 - 예) 문장 "세상에는 시계열 데이터가 참 많다"
 - 단어가 나타나는 순서가 중요
 - 샘플의 길이가 다름

■ 시계열 데이터를 인식하는 고전적인 모델

- ARIMA(autoregressive integrated moving average)와 SARIMA(seasonal ARIMA)
- Prophet 등

■ 시계열 데이터를 인식하는 딥러닝 모델

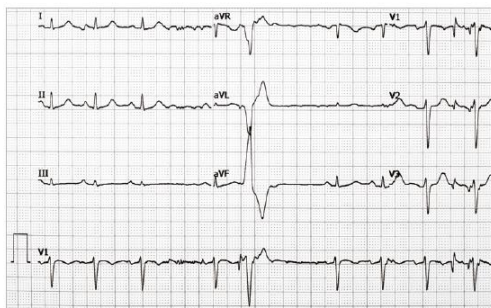
- 순환 신경망 recurrent neural network(RNN)
- LSTM_{long short-term memory}: 선별 기억 능력을 갖춰 장기 문맥 처리에 유리

8.1 시계열 데이터의 이해

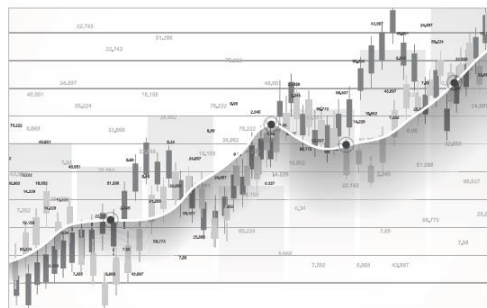
■ 시계열 데이터

- 시간 축을 따라 신호가 변하는 동적 데이터
- 앞에서 공부한 SVM, 다층 퍼셉트론, 깊은 다층 퍼셉트론, 컨볼루션 신경망
 - 정적 데이터를 한꺼번에 입력받기 때문에 시계열 데이터에 부적합
 - 시계열 데이터를 정적 데이터로 변환하여 입력하면 정보 손실이 큼

■ 딥러닝에서는 시계열 특성을 반영하는 순환 신경망 또는 LSTM 활용



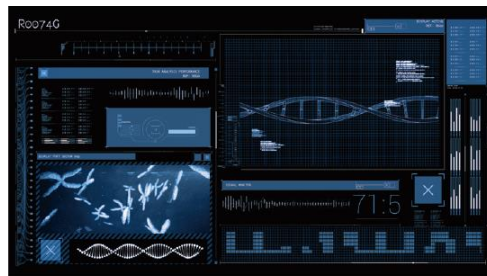
(a) 심전도



(b) 주식 시세



(c) 음성 인식 데이터



(d) 유전자 염기 서열

그림 8-1 다양한 시계열 데이터

8.1.1 시계열 데이터의 특성

■ 시계열 데이터의 독특한 특성

- 요소의 순서가 중요
 - 예) "세상에는 시계열 데이터가 참 많다" 를 "시계열 참 데이터가 많다 세상에는" 으로 바꾸면 의미 훼손
- 샘플의 길이가 다름
 - 예) 짧은 발음 "인공지능" 과 긴 발음 "인~공~~지~능"
- 문맥 의존성
 - 예) "시계열은 앞에서 말한 바와 ... 특성이 있다" 에서 "시계열은" 과 "특성이 있다" 는 밀접한 관련성
- 계절성
 - 예) 상추 판매량, 미세먼지 수치, 항공권 판매량 등

■ 시계열 데이터의 표현 $\mathbf{x} = (\mathbf{a}^1 \ \mathbf{a}^2 \cdots \mathbf{a}^t)$ (8.1)

- 가변 길이로 벡터의 벡터임
- 예) 매일 기온, 습도, 미세먼지 농도를 기록한다면, $\mathbf{a}^1 = (23.5, 42, 0.1)$, $\mathbf{a}^2 = (25.5, 45, 0.08)$, ...

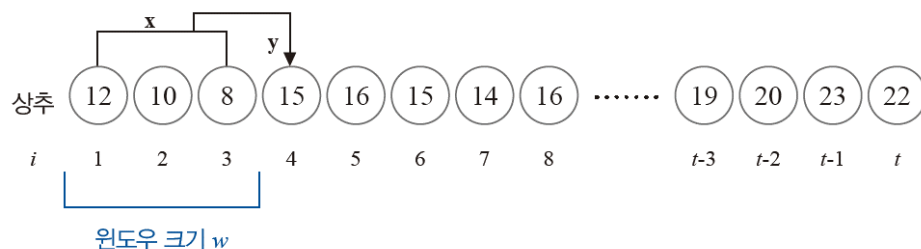
8.1.2 미래 예측을 위한 데이터 준비

- 순환 신경망은 유연한 구조라 여러 문제에 적용 가능
 - 대표적 응용은 미래 예측(prediction 또는 forecasting)
 - 내일 주가 예측
 - 내일 날씨 예측
 - 기계의 고장 예측
 - 풍속과 풍향 예측(풍력 발전기의 효율 향상)
 - 농산물 가격/수요량 예측 등
 - 언어 번역에 응용
 - 음성 인식에 응용
 - 생성 모델에 응용(예, 사진을 보고 설명 문장 생성)

8.1.2 미래 예측을 위한 데이터 준비

■ 예, 농산물 수요량 예측 문제에서 데이터

- 농산물 유통업자가 5년 동안 매일 판매량을 기록했다면 길이 $t=365*5=1825$ 인 샘플
- 하나의 긴 샘플을 가지고 어떻게 모델링하고 어떻게 미래를 예측하나?
 - 윈도우 크기(w) 단위로 패턴을 잘라 여러 개의 샘플을 수집([그림 8-2]에서는 $w=3$)
 - 얼마나 먼 미래를 예측할 지 지정하는 수평선 계수 h ([그림 8-2]에서는 $h=1$)



(a) 입력 데이터 샘플링

샘플	x	y
1	(12, 10, 8)	15
2	(10, 8, 15)	16
3	(8, 15, 16)	15
4	(15, 16, 15)	14
...
$t-w$	(19, 20, 23)	22

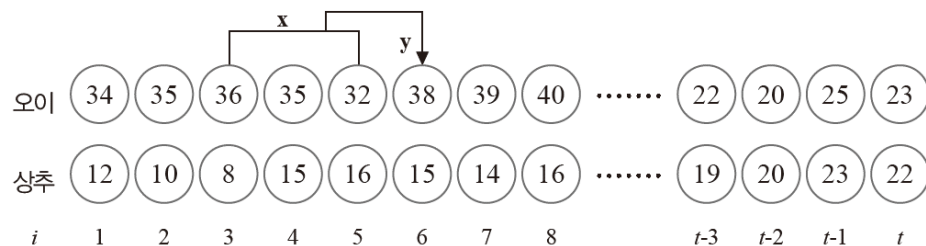
(b) 입력 패턴에서 생성한 샘플

그림 8-2 미래 예측 문제에서 샘플 생성하기(단일 채널)

8.1.2 미래 예측을 위한 데이터 준비

■ 다중 품목을 표현하는 데이터

- 벡터의 벡터 구조
- 예) 상추와 오이라는 두 품목을 동시에 고려하는 데이터



(a) 입력 패턴

샘플	x	y
1	$((34, 12), (35, 10), (36, 8))$	$(35, 15)$
2	$((35, 10), (36, 8), (35, 15))$	$(32, 16)$
3	$((36, 8), (35, 15), (32, 16))$	$(38, 15)$
4	$((35, 15), (32, 16), (38, 15))$	$(39, 14)$
...
$t - w$	$((22, 19), (20, 20), (25, 23))$	$(23, 22)$

(b) 입력 패턴에서 생성한 샘플

그림 8-3 미래 예측 문제에서 샘플 생성하기(다중 채널)

8.1.3 시계열 데이터 사례: 비트코인 가격

■ 코인데스크에서 데이터 다운로드

- www.coindesk.com

[Currency, Date, Closing Price(USD),
24h Open(USD), 24h High(USD), 24h Low(USD)]의
6개 열로 구성



	A	B	C	D	E	F	G
1	Currency	Date	Closing Pr	24h Open	24h High	24h Low (USD)	
2	BTC	2019-02-28	3772.94	3796.64	3824.17	3666.52	
3	BTC	2019-03-01	3799.68	3773.44	3879.23	3753.8	
4	BTC	2019-03-02	3811.61	3799.37	3840.04	3788.92	
5	BTC	2019-03-03	3804.42	3806.69	3819.19	3759.41	
6	BTC	2019-03-04	3782.66	3807.85	3818.7	3766.24	
7	BTC	2019-03-05	3689.86	3783.36	3804.35	3663.48	
8	BTC	2019-03-06	3832.08	3701.05	3866.72	3688.7	
9	BTC	2019-03-07	3848.96	3832.59	3881.97	3802.52	
10	BTC	2019-03-08	3859.84	3848.96	3890.75	3827.67	
11	BTC	2019-03-09	3828.37	3859.8	3918	3778.52	
12	BTC	2019-03-10	3898.87	3841.89	3948.88	3832.14	
13	BTC	2019-03-11	3899.66	3916.76	3921.93	3865.92	
14	BTC	2019-03-12	3851.25	3899.46	3913.46	3819.93	

(a) 코인데스크에서 데이터를 다운로드하기

(b) 비트코인 가격이 저장된 데이터 프레임

그림 8-4 코인데스크 사이트에서 다운로드한 비트코인 가격 데이터

TIP 책에서는 시작 날짜와 끝 날짜를 2019년 2월 28일과 2020년 2월 27일로 설정해 1년치 데이터를 다운로드했다. 독자는 가장 최근 데이터를 다운로드해 사용해도 좋고 책과 같은 기간을 설정해도 좋다.

TIP 다운로드 사이트의 화면이나 메뉴가 바뀌는 경우가 빈번하다. 책과 다른 화면이 나온다면 예제 파일로 제공하는 데이터를 사용하면 된다.

8.1.3 시계열 데이터 사례: 비트코인 가격

■ 비트코인 가격 데이터 읽기 [프로그램 8-1(a)]

- pandas 라이브러리 이용

따라해보기

프로그램 8-1(a)

비트코인 가격 데이터 읽기

```
01 import numpy as np
02 import pandas as pd
03 import matplotlib.pyplot as plt
04
05 # 코인데스크 사이트에서 다운로드한 1년치 비트코인 가격 데이터 읽기
06 f=open('BTC_USD_2019-02-28_2020-02-27-CoinDesk.csv','r')
07 coindesk_data=pd.read_csv(f,header=0)
08 seq=coindesk_data[['Closing Price (USD)']].to_numpy() # 종가만 취함
09 print('데이터 길이:',len(seq),'\n앞쪽 5개 값:',seq[0:5])
10
11 # 그래프로 데이터 확인
12 plt.plot(seq,color='red')
13 plt.title('Bitcoin Prices (1 year from 2019-02-28)')
14 plt.xlabel('Days');plt.ylabel('Price in USD')
15 plt.show()
```

첫번째 행을 헤더로 사용하라는 지시

Closing Price 행을 취하고 넘파이 배열로 변환

앞 쪽 다섯 개 샘플을 취함

8.1.3 시계열 데이터 사례: 비트코인 가격

데이터 길이: 365

앞쪽 5개 값: [[3772.93633533]

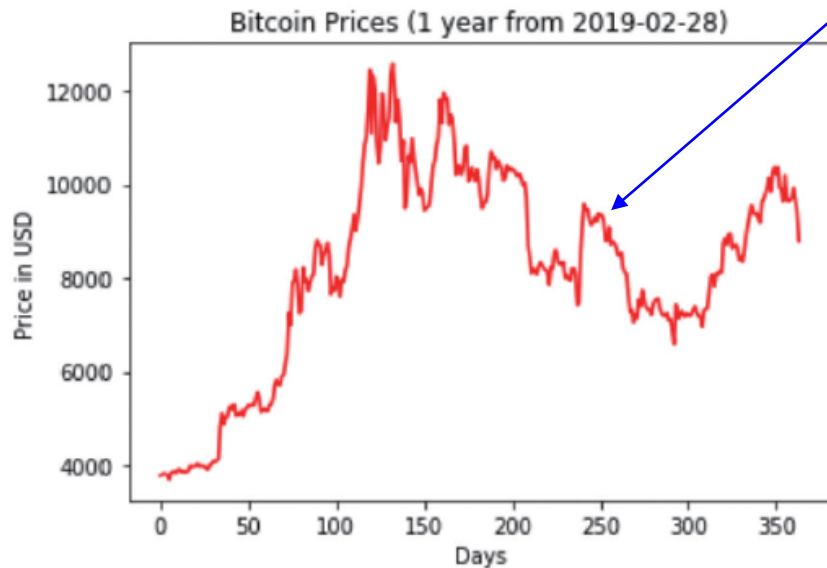
[3799.67854295]

[3811.61197937]

[3804.41917011]

[3782.66410112]]

심한 등락. 특히 전반부에 치솟음.
후반부는 상대적으로 안정세



8.1.3 시계열 데이터 사례: 비트코인 가격

NOTE [[]]와 []의 차이

[프로그램 8-1(a)]에서 08행의 `coindesk_data[['Closing Price (USD)']]` 구문에 `[[...]]`을 사용했다. 08행을 `coindesk_data['Closing Price (USD)']`로 바꾸어 `[...]` 방식을 사용하면 무슨 차이가 있을까? 간단한 예를 들면, `[[...]]` 방식은 `[[3] [5] [7] [6]]`과 같이 표현되고 `[...]` 방식은 `[3 5 7 6]`으로 표현된다. [프로그램 8-1(a)]는 매일 형성되는 네 가지 주식 시세인 종가, 시가, 고가, 저가 중에서 종가만 취하여 실험을 하기 때문에 벡터의 벡터인 `[[...]]`와 벡터인 `[...]`의 차이가 두드러지지 않는다. 네 값을 모두 사용하는 다중 채널의 경우는 반드시 벡터의 벡터인 `[[...]]` 표현을 사용해야 한다. 종가만 사용하는 프로그램에서 `[[...]]` 표현으로 코딩해두면 다중 채널로 확장이 쉬워지기 때문에 더 좋은 프로그램이 된다. 다중 채널로 확장하는 작업은 [프로그램 8-3]에서 다룬다.

8.1.3 시계열 데이터 사례: 비트코인 가격

■ 윈도우 단위로 잘라 샘플링하는 [프로그램 8-1(b)]

윈도우 크기
30으로 변경 해보기

프로그램 8-1(b) 1년치 비트코인 가격 데이터를 윈도우로 자르기([그림 8-2])

```
16
17 # 시계열 데이터를 윈도우 단위로 자르는 함수
18 def seq2dataset(seq,window,horizon):
19     X=[]; Y=[]
20     for i in range(len(seq)-(window+horizon)+1):
21         x=seq[i:(i+window)]
22         y=(seq[i+window+horizon-1])
23         X.append(x); Y.append(y)
24     return np.array(X), np.array(Y)
25
26 w=7    # 윈도우 크기
27 h=1    # 수평선 계수
28
29 X,Y = seq2dataset(seq,w,h)
30 print(X.shape,Y.shape)
31 print(X[0],Y[0]); print(X[-1],Y[-1])
```

맨 앞과 맨 뒤 샘플을 출력

8.1.3 시계열 데이터 사례: 비트코인 가격

(358, 7, 1) (358, 1)
[[3772.93633533]
[3799.67854295]
[3811.61197937]
[3804.41917011]
[3782.66410112]
[3689.86289319]
[3832.08088473]] [3848.95636968]
[[9701.0371915]
[9631.48494596]
[9670.85865437]
[9689.08674285]
[9919.55144784]
[9640.46950506]
[9392.86962872]] [8787.97836316]

8.2 순환 신경망

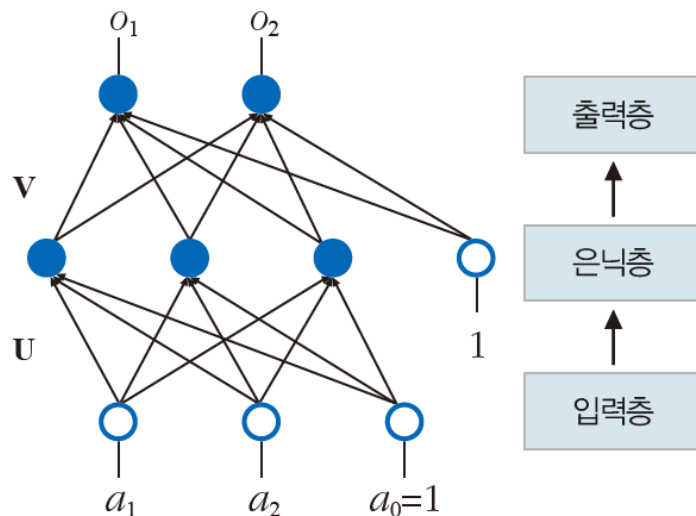
■ 시계열 데이터를 처리하는 신경망

- 시간에 따라 값이 하나씩 순차적으로 들어온다는 사실을 반영하는 신경망을 설계해야 함
- 다행히 다층 퍼셉트론을 약간 고쳐서 사용하면 됨

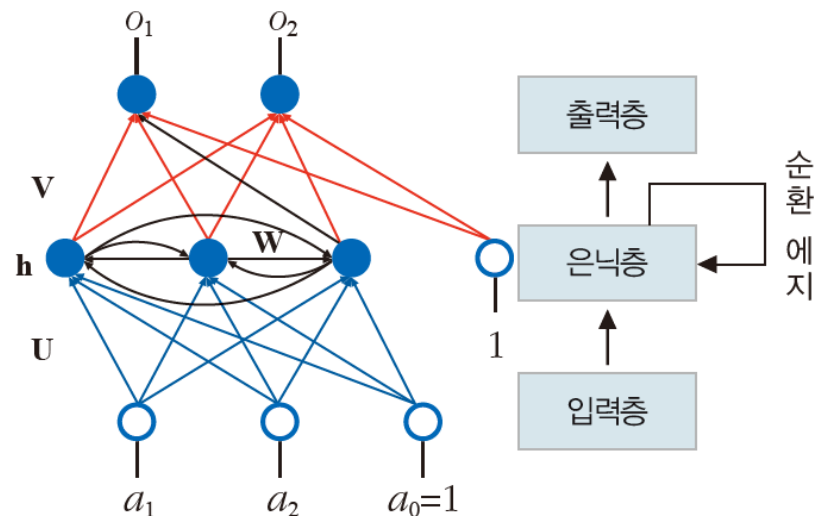
8.2.1 구조와 동작, 학습 알고리즘

■ 순환 신경망의 구조([그림 8-5(b)])

- 은닉층 노드 사이에 에지(순환 에지)가 있다는 사실을 제외하고 다층 퍼셉트론과 동일함



(a) 다층 퍼셉트론([그림 4-13])



(b) 순환 신경망([그림 4-13]에 순환 에지 추가)

그림 8-5 다층 퍼셉트론과 순환 신경망의 비교

- 가중치 집합
 - 다층 퍼셉트론 $\{U, V\}$
 - 순환 신경망 $\{U, V, W\}$

8.2.1 구조와 동작, 학습 알고리즘

■ 순환 신경망에 데이터 입력([그림 8-6])

- 펼쳐서 그리면 이해가 쉬움
- i 순간에 \mathbf{a}^i 가 입력됨
- [그림 8-5(b)]의 a_0, a_1, a_2 가 \mathbf{a}^i 를 구성

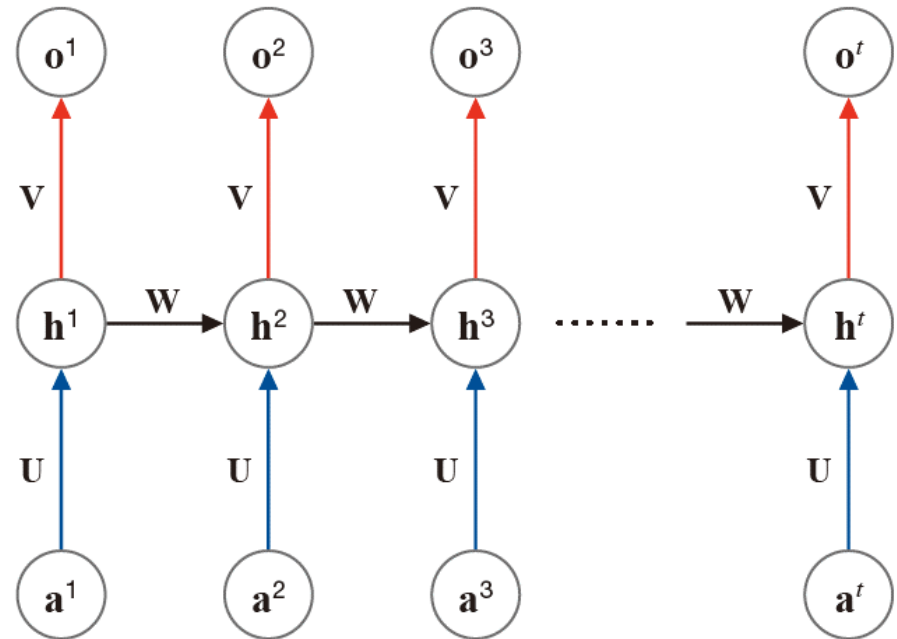


그림 8-6 펼친 순환 신경망

- 가중치 공유
 - 순간마다 서로 다른 가중치를 가지는 것이 아님
 - 모든 순간이 $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$ 를 공유

8.2.1 구조와 동작, 학습 알고리즘

■ 순환 신경망의 동작

- i 순간의 \mathbf{a}^i 는 가중치 \mathbf{U} 를 통해 은닉층의 상태 \mathbf{h}^i 에 영향을 미치고, \mathbf{h}^i 는 가중치 \mathbf{V} 를 통해 출력값 \mathbf{o}^i 에 영향을 미침. \mathbf{h}^{i-1} 는 가중치 \mathbf{W} 를 통해 \mathbf{h}^i 에 영향을 미침

- 은닉층에서 일어나는 계산

$$\mathbf{h}^i = \tau_1(\mathbf{W}\mathbf{h}^{i-1} + \mathbf{U}\mathbf{a}^i) \quad (8.2)$$

- 출력층에서 일어나는 계산

$$\mathbf{o}^i = \tau_2(\mathbf{V}\mathbf{h}^i) \quad (8.3)$$

이 항을 제외하면 다층 퍼셉트론과 동일

- 식 (8.2)에서 $\mathbf{W}\mathbf{h}^{i-1}$ 항을 제외하면 다층 퍼셉트론과 동일

- 순환 신경망은 이 항을 통해 이전 순간의 은닉층 상태 \mathbf{h}^{i-1} 를 현재 순간의 은닉층 상태 \mathbf{h}^i 로 전달하여 시간성을 처리함

8.2.1 구조와 동작, 학습 알고리즘

■ 순환 신경망의 학습

- 학습 알고리즘은 최적의 $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$ 를 알아냄
- BPTT(back-propagation through time) 알고리즘

TIP 순환 신경망의 학습 알고리즘을 BPTT(Back-Propagation Through Time)라고 한다. 자세한 내용을 알고자 하는 독자는 『기계 학습(오일석, 한빛아카데미)』의 8.2.3항을 참조한다.

8.2.2 선별 기억력을 갖춘 LSTM

■ 순환 신경망의 기억력 한계

- 은닉층 상태를 다음 순간으로 넘기는 기능을 통해 과거를 기억
- 하지만 장기 문맥 의존성(멀리 떨어진 요소가 밀접한 상호작용하는 현상)을 제대로 처리하지 못하는 한계
- 계속 들어오는 입력의 영향으로 기억력 감퇴
- 사람은 선별 기억 능력으로 오래 전 기억을 간직함

8.2.2 선별 기억력을 갖춘 LSTM

■ LSTM은 게이트라는 개념으로 선별 기억 확보

- \circ 는 열림, \times 는 닫힘
- 실제로는 게이트는 0~1 사이의 실수값으로 열린 정도를 조절
- 게이트의 여닫는 정도는 가중치로 표현되며 가중치는 학습으로 알아냄

■ LSTM의 가중치

- 순환 신경망의 $\{U, V, W\}$ 에 4개를 추가하여 $\{U, U^i, U^o, W, W^i, W^o, V\}$
- i 는 입력 게이트, o 는 출력 게이트

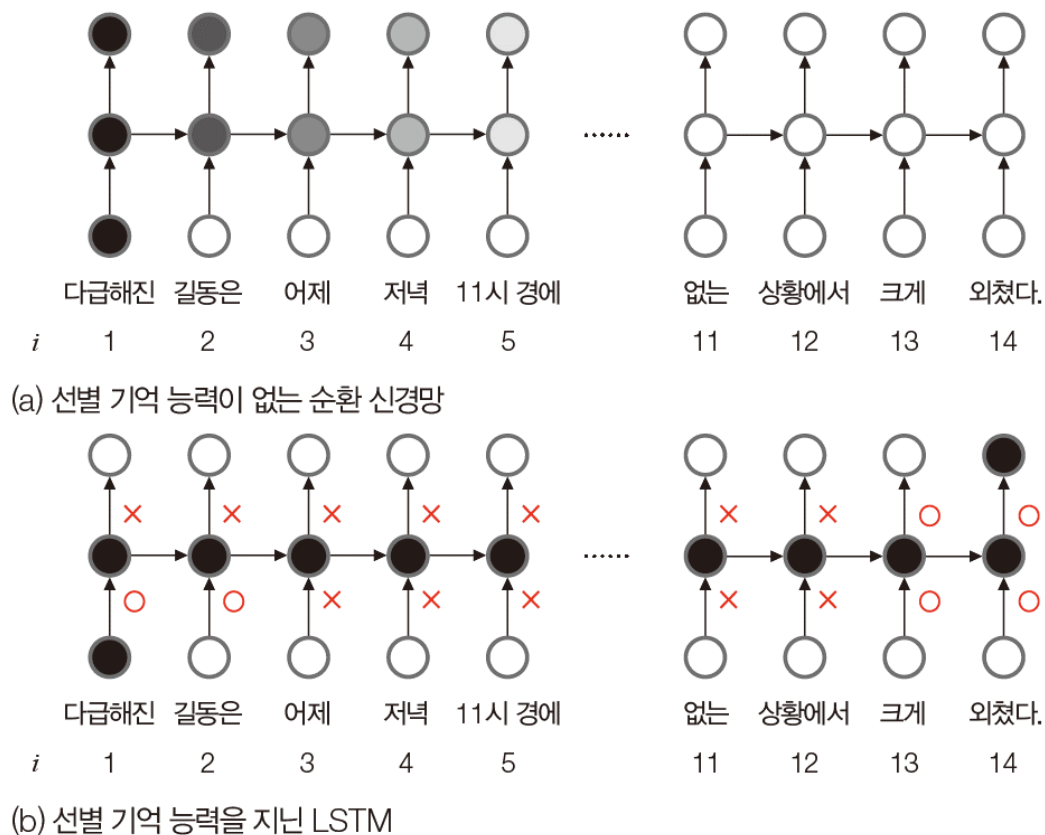
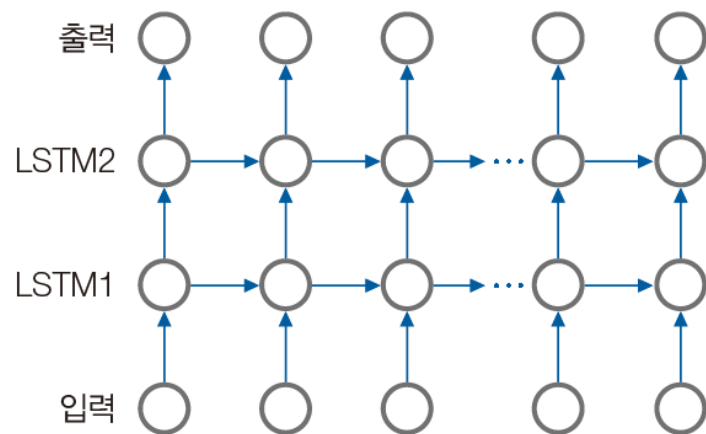


그림 8-7 순환 신경망과 LSTM

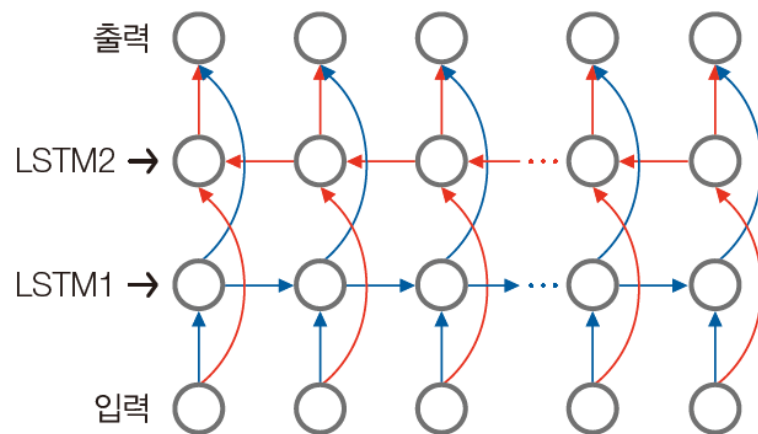
8.2.3 LSTM의 유연한 구조

■ 다양한 구조 설계 가능

- 양방향으로 문맥을 살필 필요가 있는 경우는 양방향 LSTM 사용
 - 예) "잘 달리는 이 차는 ..."과 "고산지대에서 생산한 이 차는 향기가 좋다"에서 앞 문장은 왼쪽, 뒤 문장은 오른쪽 단어를 보고 각각 car와 tea로 번역



(a) 적층 LSTM

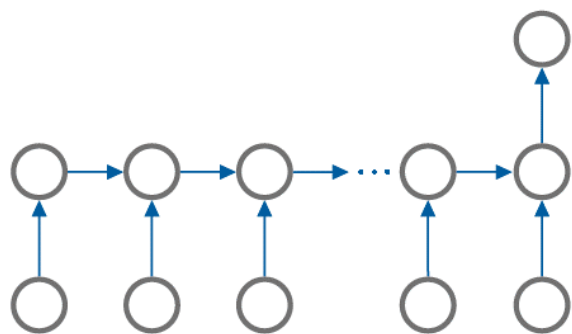


(b) 양방향 LSTM

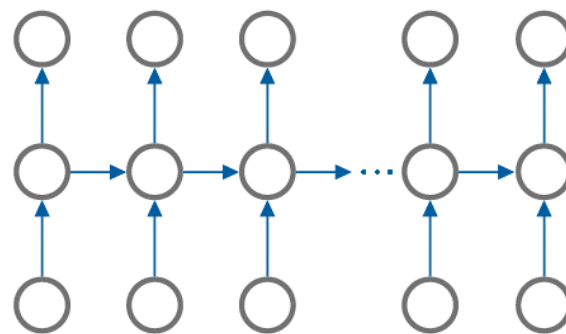
그림 8-8 적층 LSTM과 양방향 LSTM

8.2.3 LSTM의 유연한 구조

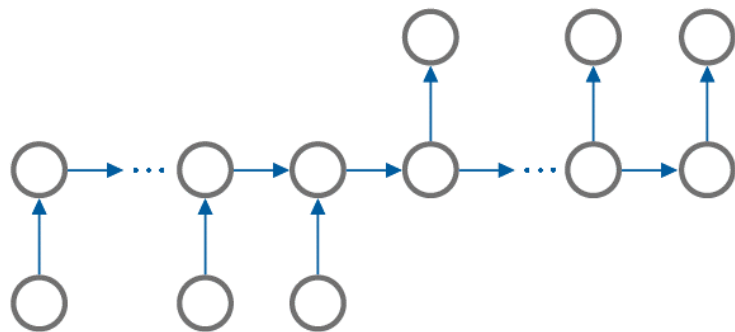
■ 응용 문제에 따른 다양한 구조



(a) 분류 또는 예측 문제



(b) 비디오 프레임 분류



(c) 언어 번역

그림 8-9 다양한 문제에 적용할 수 있는 LSTM

8.3 LSTM으로 시계열 예측하기

- 시계열 데이터를 보고 미래를 예측하는 프로그래밍 실습
 - 단일 채널
 - 종가만 고려하는 [프로그램 8-2]
 - 다중 채널
 - 종가, 시가, 고가, 저가를 모두 고려하는 [프로그램 8-3]

8.3.1 단일 채널 비트코인 가격 예측

■ 7일치 가격을 보고 내일 가격을 예측하는 LSTM 프로그래밍

프로그램 8-2 LSTM을 이용한 비트코인 가격 예측

```

01 import numpy as np
02 import pandas as pd
03 import matplotlib.pyplot as plt
04
05 # 코인데스크 사이트에서 1년치 비트코인 가격 데이터 읽기
06 f=open("BTC_USD_2019-02-28_2020-02-27-CoinDesk.csv","r")
07 coindesk_data=pd.read_csv(f,header=0)
08 seq=coindesk_data[['Closing Price (USD)']].to_numpy() # 종가만 취함
09
10 # 시계열 데이터를 윈도우 단위로 자르는 함수
11 def seq2dataset(seq>window,horizon):
12     X=[]; Y=[]
13     for i in range(len(seq)-(window+horizon)+1):
14         x=seq[i:(i+window)]
15         y=(seq[i+window+horizon-1])
16         X.append(x); Y.append(y)
17     return np.array(X), np.array(Y)
18
19 w=7 # 윈도우 크기
20 h=1 # 수평선 계수
21
22 X,Y=seq2dataset(seq,w,h)
23

```

01~22행:
데이터 읽고 샘플링하는 [프로그램 8-1]

8.3.1 단일 채널 비트코인 가격 예측

```

24 from tensorflow.keras.models import Sequential
25 from tensorflow.keras.layers import Dense, LSTM, Dropout
26
27 # 훈련 집합과 테스트 집합으로 분할
28 split=int(len(X)*0.7)
29 x_train=X[0:split]; y_train=Y[0:split]
30 x_test=X[split:]; y_test=Y[split:]
31
32 # LSTM 모델 설계와 학습
33 model=Sequential()
34 model.add(LSTM(units=128,activation='relu',input_shape=x_train[0].shape))
35 model.add(Dense(1))
36 model.compile(loss='mae',optimizer='adam',metrics=['mae'])
37 hist=model.fit(x_train,y_train,epochs=200,batch_size=1,validation_data=
(x_test,y_test),verbose=2)
38
39 # LSTM 모델 평가
40 ev=model.evaluate(x_test,y_test,verbose=0)
41 print("손실 함수:",ev[0],"MAE:",ev[1])
42
43 # LSTM 모델로 예측 수행
44 pred=model.predict(x_test)
45 print("평균절댓값백분율오차(MAPE):",sum(abs(y_test-pred)/y_test)/len(x_test))

```

LSTM 층을 위한 함수 불러옴
 출력층 은닉층
 입력층
 손실함수로 mae 사용
 출력층의 activation 매개변수를 생략하여 기본값인 선형 사용 (출력이 가격(연속값)이어서 회귀 문제이기 때문)

8.3.1 단일 채널 비트코인 가격 예측

```
46
47 # 학습 곡선
48 plt.plot(hist.history['mae'])
49 plt.plot(hist.history['val_mae'])
50 plt.title('Model mae')
51 plt.ylabel('mae')
52 plt.xlabel('Epoch')
53 plt.ylim([120,800])
54 plt.legend(['Train','Validation'], loc='best')
55 plt.grid()
56 plt.show()
57
58 # 예측 결과 시각화
59 x_range=range(len(y_test))
60 plt.plot(x_range,y_test[x_range], color='red')
61 plt.plot(x_range,pred[x_range], color='blue')
62 plt.legend(['True prices','Predicted prices'], loc='best')
63 plt.grid()
64 plt.show()
65
66 # 일부 구간을 확대하여 시각화
67 x_range=range(50,64)
68 plt.plot(x_range,y_test[x_range], color='red')
69 plt.plot(x_range,pred[x_range], color='blue')
70 plt.legend(['True prices','Predicted prices'], loc='best')
71 plt.grid()
72 plt.show()
```

8.3.1 단일 채널 비트코인 가격 예측

Train on 249 samples, validate on 108 samples

Epoch 1/200

249/249 - 3s - loss: 1858.8645 - mae: 1858.8646 - val_loss: 447.8866 - val_mae: 447.8866

...

Epoch 200/200

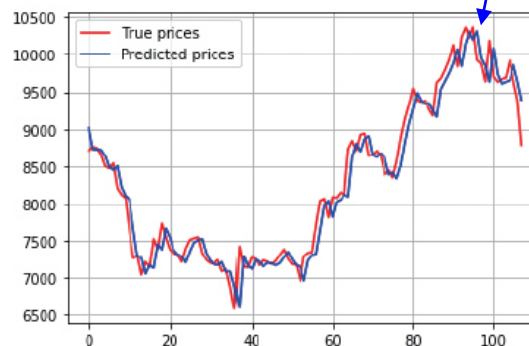
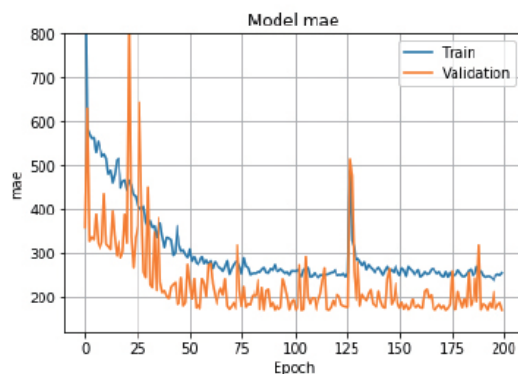
249/249 - 1s - loss: 253.9756 - mae: 253.9756 - val_loss: 168.5357 - val_mae: 168.5357

손실 함수: 169.59739854600696 MAE: 169.5974

평균절댓값백분율오차(MAPE): [0.02028812]

MAE
MAPE

얼핏 아주 훌륭한 예측이라 보임



확대해 보면 파란 곡선(예측 값)은 빨간 곡선(정답)을 오른쪽으로 한 칸 이동한 모양. 무엇을 뜻하나?

8.3.1 단일 채널 비트코인 가격 예측

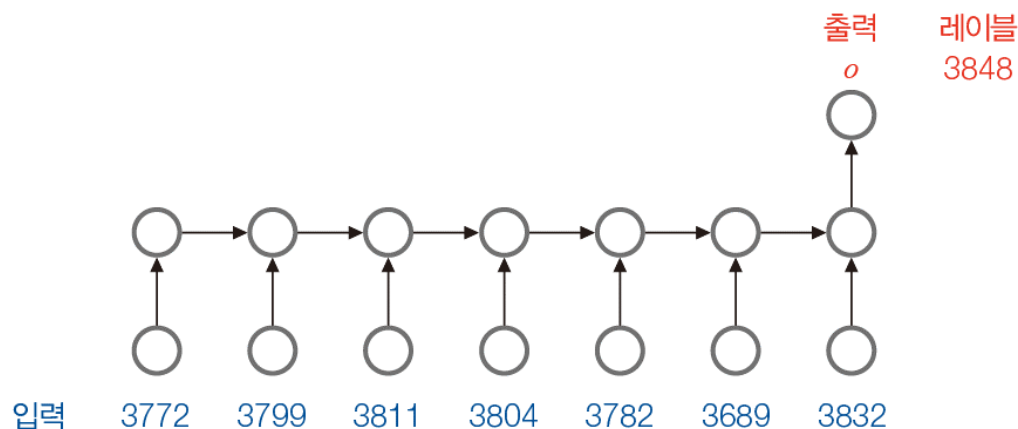


그림 8-10 [프로그램 8-2]의 신경망에 샘플 입력 후 출력과 레이블의 오차를 구하는 과정

NOTE 데이터 편향

[프로그램 8-2]의 실행 결과로 얻은 학습 곡선을 보면 검증 집합의 MAE가 훈련 집합보다 작은 기이한 현상을 볼 수 있다. [프로그램 8-1(a)]가 출력한 비트코인 가격의 원본 데이터를 보면, 초반부에 심한 등락이 나타나고 후반부는 상대적으로 안정된 추세를 보인다. 데이터를 7:3으로 나눌 때 앞부분 70%는 훈련 집합, 나머지 뒷부분은 테스트 집합에 담았다. 따라서 원천적으로 훈련 집합의 예측은 테스트 집합보다 어렵기 때문에 테스트 집합의 MAE가 더 작은 현상이 발생했다. 이처럼 데이터의 패턴이 한쪽으로 치우친 현상을 데이터 편향(data bias)이라고 한다. 의사와 간호사를 인식하는 인공지능 시스템을 학습했는데, 나중에 남자 간호사 사진을 입력하면 의사라고 분류하고 여자 의사 사진을 입력하면 간호사로 분류하는 현상도 데이터 편향 때문에 나타난다.

8.3.2 성능 평가

■ 성능 기준

- 평균절댓값오차(mean absolute error(MAE): 스케일 문제에 대처하지 못함

$$\text{평균절댓값오차: MAE} = \frac{1}{|M|} \sum_{x \in M} |y - o| \quad (8.4)$$

- 평균절댓값백분율오차: 스케일 문제에 대처

$$\text{평균절댓값백분율오차: MAPE} = \frac{1}{|M|} \sum_{x \in M} \left| \frac{y - o}{y} \right| \quad (8.5)$$

표 8-1 평균절댓값오차와 평균절댓값백분율오차

데이터 스케일	데이터(y_test는 참값, pred는 예측값)	평균절댓값오차(MAE)	평균절댓값백분율오차(MAPE)
두 자릿수	y_test [12 20 18 22 28]	(2+1+2+2+2)/5 =1.8	(2/12+1/20+2/18+2/22+2/28)/5 =0.0980
	pred [10 21 16 20 26]		
세 자릿수	y_test [120 200 180 220 280]	(20+10+20+20+20)/5 =18	(20/120+10/200+20/180+20/220+20/280)/5=0.0980
	pred [100 210 160 200 260]		

8.3.2 성능 평가

■ 성능 기준

■ 등락 정확률

- 등락을 얼마나 정확하게 맞히는지 측정
- 맞힌 경우의 수를 전체 샘플 수로 나눔

표 8-2 등락 정확률

샘플 i	x_test[i]	y_test[i]	pred[i]	맞힘
1	[.,, .., .., 21]	23	24	o
2	[.,, .., .., 25]	20	26	x
3	[.,, .., .., 22]	24	23	o
4	[.,, .., .., 28]	25	26	o
5	[.,, .., .., 21]	18	17	o
6	[.,, .., .., 32]	31	33	x
7	[.,, .., .., 35]	36	37	o
8	[.,, .., .., 20]	19	22	x
				등락 정확률 = $5/8 = 62.5\%$

8.3.3 다중 채널 비트코인 가격 예측

■ 다중 채널을 사용하여 예측하는 [프로그램 8-3]

프로그램 8-3 다중 채널 비트코인 가격 예측

```

01 import numpy as np
02 import pandas as pd
03 import matplotlib.pyplot as plt
04
05 # 코인데스크 사이트에서 1년치 비트코인 가격 데이터 읽기
06 f=open("BTC_USD_2019-02-28_2020-02-27-CoinDesk.csv","r")
07 coindesk_data=pd.read_csv(f,header=0)
08 seq=coindesk_data[['Closing Price (USD)', '24h Open (USD)', '24h High (USD)', '24h
    Low (USD)']].to_numpy()      # 증가, 시가, 고가, 저가를 모두 취함
09
10 # 시계열 데이터를 윈도우 단위로 자르는 함수
11 def seq2dataset(seq,window,horizon):
12     X=[]; Y=[]
13     for i in range(len(seq)-(window+horizon)+1):
14         x=seq[i:(i+window)]
15         y=(seq[i+window+horizon-1])
16         X.append(x); Y.append(y)
17     return np.array(X), np.array(Y)
18
19 w=7      # 윈도우 크기
20 h=1      # 수평선 계수
21
22 X,Y = seq2dataset(seq,w,h)
23 print(X.shape,Y.shape)
24 print(X[0],Y[0])

```

(증가,시가,고가,저가)를 데이터로 사용

← 0번 샘플 확인

8.3.3 다중 채널 비트코인 가격 예측

```

25
26 from tensorflow.keras.models import Sequential
27 from tensorflow.keras.layers import Dense, LSTM, Dropout
28
29 # 훈련 집합과 테스트 집합으로 분할
30 split=int(len(X)*0.7)
31 x_train=X[0:split]; y_train=Y[0:split]
32 x_test=X[split:]; y_test=Y[split:]
33
34 # LSTM 모델의 설계와 학습
35 model = Sequential()
36 model.add(LSTM(units=128,activation='relu',input_shape=x_train[0].shape))
37 model.add(Dense(4))
38 model.compile(loss='mae',optimizer='adam',metrics=['mae'])
39 hist=model.fit(x_train,y_train,epochs=200,batch_size=1,validation_data=
    (x_test,y_test),verbose=2)
40
41 # LSTM 모델 평가
42 ev=model.evaluate(x_test,y_test,verbose=0)
43 print("손실 함수:",ev[0],"MAE:",ev[1])
44
45 # LSTM 모델로 예측 수행
46 pred=model.predict(x_test)
47 print("LSTM 평균절대값백분율오차(MAPE):",sum(abs(y_test-pred)/y_test)/len(x_test))
48
49 # 학습 곡선
50 plt.plot(hist.history['mae'])
51 plt.plot(hist.history['val_mae'])
52 plt.title('Model mae')
53 plt.ylabel('mae')
54 plt.xlabel('Epoch')
55 plt.ylim([100,600])
56 plt.legend(['Train','Validation'], loc='best')
57 plt.grid()
58 plt.show()

```

출력층의 노드 개수는 4개

8.3.3 다중 채널 비트코인 가격 예측

훈련 집합 X와 Y의 텐서 모양

(358, 7, 4) (358, 4)

```
[[3772.93633533 3796.63728431 3824.16587937 3666.52401643]
 [3799.67854295 3773.44146075 3879.23118467 3753.80002246]
 [3811.61197937 3799.36702601 3840.04482307 3788.91849833]
 [3804.41917011 3806.69151279 3819.19435612 3759.40921647]
 [3782.66410112 3807.84575592 3818.69548135 3766.24204823]
 [3689.86289319 3783.35506344 3804.35361623 3663.47774336]
 [3832.08088473 3701.04987103 3866.71870424 3688.69715385]]
 [3848.95636968 3832.59242908 3881.96576977 3802.51605364]
```

Train on 249 samples, validate on 108 samples

Epoch 1/200

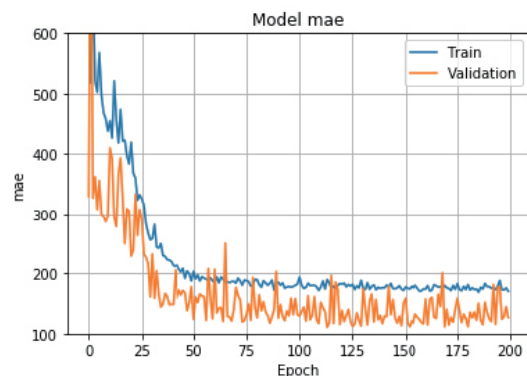
249/249 - 3s - loss: 1527.8208 - mae: 1527.8209 - val_loss: 329.5129 - val_mae: 329.5129
...

Epoch 200/200

249/249 - 1s - loss: 170.3932 - mae: 170.3933 - val_loss: 126.5666 - val_mae: 126.5666

손실 함수: 126.56658257378473 MAE: 126.56658

LSTM 평균절댓값백분율오차(MAPE): [0.02284203 0.00305788 0.01866189 0.01706315]



(종가,시가,고가,저가)의 MAPE

8.5 자연어 처리

■ 언어를 사용하는 인간

- 다른 동물보다 지능이 월등하다는 증거
- 시를 읽고 반응하는 인간

연탄재 함부로 발로 차지 마라

너는

누구에게 한 번이라도 뜨거운 사람이었느냐

■ 자연어 처리_{natural language processing(NLP)}

- 인간이 구사하는 언어를 자동으로 처리하는 인공지능 분야
- 다양한 응용
 - 언어 번역
 - 영화평 댓글 분석하여 흥행 추정
 - 고객 응대 챗봇
 - 소설이나 시를 쓰는 창작 인공지능 등

8.5.1 텍스트 데이터에 대한 이해

■ 영화평 데이터셋인 IMDB의 예제 문장

- 텍스트의 특성이 잘 나타남

Once again Mr. Costner has dragged out a movie for far longer than necessary. Aside from the terrific sea rescue sequences, of which there are very few I just did not care about any of the characters. Most of us have ghosts in the closet, and Costner's character are realized early on, and then forgotten until much later, by which time I did not care,

■ 텍스트 데이터의 특성

- 시계열 데이터로서 시간 정보가 있고 샘플마다 길이가 다르다는 기본 성질
- 그 외 독특한 특성
 - 심한 잡음
 - 형태소 분석 필요
 - 구문론과 의미론
 - 다양한 언어 특성
 - 신경망에 입력하려면 기호를 수치로 변환해야 함

8.5.1 텍스트 데이터에 대한 이해

■ 원핫 코드 표현으로 변환하는 절차

- 예) 말뭉치_{corpus} 사례

[말뭉치]

```
Freshman loves python.  
We teach python to freshman.  
How popular is Python?
```

- 단어 수집(괄호 속은 빈도수)
 - Python(3), freshman(2), loves(1), we(1), teach(1), to(1), how(1), popular(1), is(1)
- 파이썬의 자료구조인 딕셔너리를 이용한 표현(빈도수에 따른 순위 부여)
 - {'python':1, 'freshman':2, 'loves':1, 'we':1, 'teach':1, 'to':1, 'how':1, 'popular':1, 'is':1}
- 텍스트를 숫자 코드로 변환

Freshman loves python.	→	[2 3 1]
We teach python to freshman.	→	[4 5 1 6 2]
How popular is Python?	→	[7 8 9 1]

8.5.1 텍스트 데이터에 대한 이해

■ 원핫 코드 표현

```
[231]    →  [[010000000] [001000000] [100000000]]  
[45162] →  [000100000] [000010000] [100000000] [000001000] [010000000]  
[7891]   →  [000000100] [000000010] [000000001] [100000000]
```

■ 원핫 코드의 문제점과 해결책

- 사전 크기가 크면 원핫 코드는 희소 벡터가 되어 메모리 낭비
- 단어 사이의 연관 관계를 반영하지 못함
 - 예, king과 queen, bridegroom과 bride의 관계 표현 불가능
- 8.5.3항에서는 현대적인 표현 기법인 단어 임베딩을 다룸

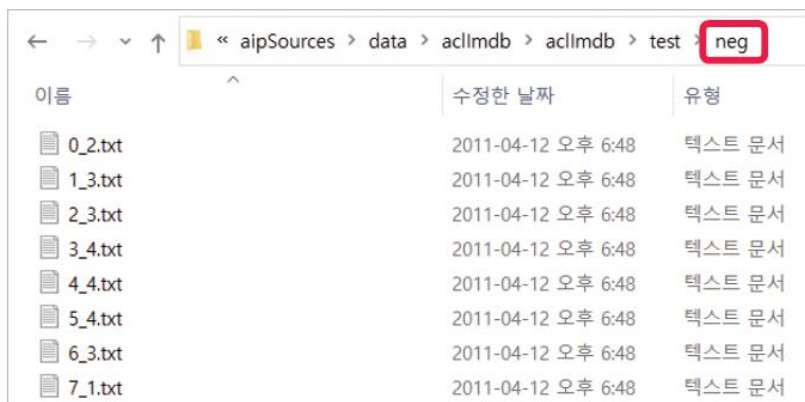
8.5.2 텍스트 데이터 사례: 영화평 데이터셋 IMDB

- 텐서플로가 제공하는 텍스트 데이터
 - 영화를 평가한 댓글을 모아둔 IMDB 데이터셋
 - 50000개의 댓글을 긍정 평가와 부정 평가로 레이블링
 - 감정 분류 sentiment classification 문제에 주로 사용
 - 다양한 토픽의 뉴스를 모아둔 Reuters 데이터셋
 - 로이터 통신의 뉴스 11228개를 46개 토픽으로 레이블링
 - 토픽 분류 문제에 주로 사용
- 이 절에서는 IMDB로 프로그래밍 실습

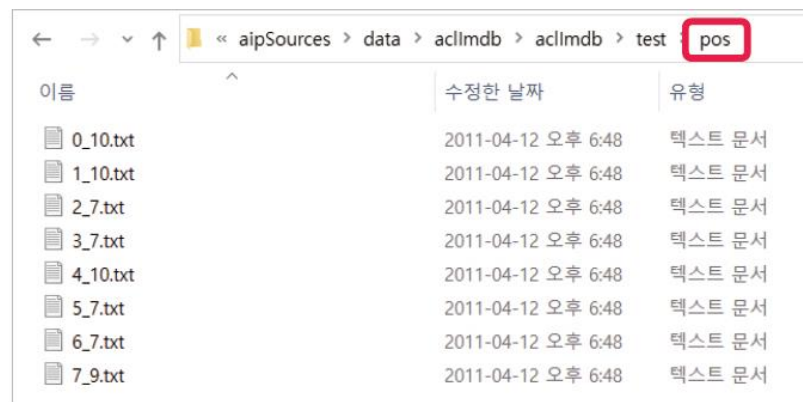
8.5.2 텍스트 데이터 사례: 영화평 데이터 셋 IMDB

■ IMDB로 실습

- <http://mng.bz/0tlo>에 접속하여 원본 데이터 다운로드
- 소스 프로그램이 있는 폴더에 data라는 폴더 만들어 거기에 저장



이름	수정한 날짜	유형
0_2.txt	2011-04-12 오후 6:48	텍스트 문서
1_3.txt	2011-04-12 오후 6:48	텍스트 문서
2_3.txt	2011-04-12 오후 6:48	텍스트 문서
3_4.txt	2011-04-12 오후 6:48	텍스트 문서
4_4.txt	2011-04-12 오후 6:48	텍스트 문서
5_4.txt	2011-04-12 오후 6:48	텍스트 문서
6_3.txt	2011-04-12 오후 6:48	텍스트 문서
7_1.txt	2011-04-12 오후 6:48	텍스트 문서



이름	수정한 날짜	유형
0_10.txt	2011-04-12 오후 6:48	텍스트 문서
1_10.txt	2011-04-12 오후 6:48	텍스트 문서
2_7.txt	2011-04-12 오후 6:48	텍스트 문서
3_7.txt	2011-04-12 오후 6:48	텍스트 문서
4_10.txt	2011-04-12 오후 6:48	텍스트 문서
5_7.txt	2011-04-12 오후 6:48	텍스트 문서
6_7.txt	2011-04-12 오후 6:48	텍스트 문서
7_9.txt	2011-04-12 오후 6:48	텍스트 문서

그림 8-16 IMDB 데이터셋의 폴더 구조 - neg와 pos 폴더

8.5.2 텍스트 데이터 사례: 영화평 데이터 셋 IMDB

■ IMDB 원본 데이터 살펴보는 [프로그램 8-6]

프로그램 8-6

IMDB 원본 텍스트 데이터 살펴보기

```
01 import os
02
03 # 원본 IMDB 데이터 읽기
04 directory='./data/aclImdb/aclImdb/test'
05 x=[];y=[]
06 for c in ['neg','pos']:
07     curr=os.path.join(directory,c)
08     for fname in os.listdir(curr):
09         if fname[-4:]==' .txt':
10             f=open(os.path.join(curr,fname),encoding='utf8')
11             x.append(f.read())
12             y.append(c)
13
14 print("첫 번째 댓글:",x[0]); print("첫 번째 평가:",y[0])
15 print("마지막 댓글:",x[-1]); print("마지막 평가:",y[-1])
```

'neg'와 'pos' 폴더 각각에서 데이터를 읽어옴

8.5.2 텍스트 데이터 사례: 영화평 데이터 셋 IMDB

첫 번째 댓글: Once again Mr. Costner has dragged out a movie for far longer than necessary. Aside from the terrific sea rescue sequences, of which there are very few I just did not care about any of the characters. Most of us have ghosts in the closet, and Costner's character are realized early on, and then forgotten until much later, by which time I did not care. ...

첫 번째 평가: neg

마지막 댓글: When I saw this movie for the first time I was both surprised and a little shocked by the blatant vibrance of the story. It is a very artistic drama with incredible special effects, spectacular acting, not to mention a very excellent job in the makeup department. ...

마지막 평가: pos

8.5.2 텍스트 데이터 사례: 영화평 데이터셋 IMDB

■ 텐서플로가 제공하는 IMDB

- 텐서플로는 쓰기 좋게 가공한 IMDB 데이터셋을 제공

프로그램 8-7(a)

텐서플로가 제공하는 IMDB 데이터 살펴보기

```
01 from tensorflow.keras.datasets import imdb
02 from tensorflow.keras.models import Sequential
03 from tensorflow.keras.layers import Dense, Flatten, Embedding
04 from tensorflow.keras import preprocessing
05
06 dic_siz=10000      # 사전의 크기(사전에 있는 단어 개수)
07 sample_siz=512    # 샘플의 크기
08
09 # tensorflow가 제공하는 간소한 버전의 IMDB 읽기
10 (x_train,y_train),(x_test,y_test)=imdb.load_data(num_words=dic_siz)
11 print(x_train.shape,x_test.shape)
12 print(x_train[0])
13
14 # 단어를 숫자, 숫자를 단어로 변환하는데 쓰는 표(표는 딕셔너리로 구현)
15 word2id=imdb.get_word_index()
16 id2word={word:id for id,word in word2id.items()}
17
18 for i in range(1,21):    상위 빈도 20개 단어 출력
19     print(id2word[i],end='/')
```

8.5.2 텍스트 데이터 사례: 영화평 데이터 셋 IMDB

훈련 집합과 테스트 집합의 크기

(25000,) (25000,)

훈련 집합의 첫 번째 샘플

[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178, 32]

the/and/a/of/to/is/br/in/it/i/this/that/was/as/for/with/movie/but/film/on/

상위 빈도 20개 단어

8.5.3 단어 임베딩

- 텍스트 데이터를 신경망에 입력하고 표현하는 방법
 - 원핫 코드 – 몇 가지 단점이 있음
 - 대안은 단어 임베딩_{word embedding}

8.5.3 단어 임베딩

■ 단어 임베딩이란?

- 단어를 저차원 공간의 벡터로 표현하는 기법
 - 보통 수백 차원을 사용
 - 밀집 벡터임
 - 단어의 의미를 표현
 - 신경망 학습을 통해 알아냄

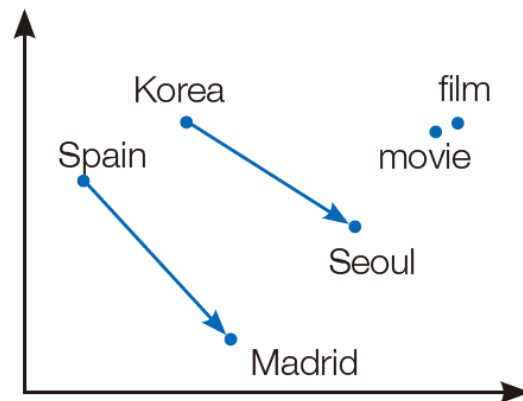


그림 8-17 단어 임베딩의 원리

[원핫 코드]

사전 크기([프로그램 8-7(a)]의 dic_siz)가 10000인 경우

"python" → $\overbrace{(1 \ 0 \ 0 \ 0 \ 0 \ \dots \ 0 \ 0 \ 0 \ \dots \ 0 \ 0 \ 0)}^{10000\text{개 요소}}$

[단어 임베딩]

단어 임베딩 공간의 차원([프로그램 8-7(b)]의 embed_space_dim)이 16인 경우

"python" → $\overbrace{(0.01 \ 0.23 \ 0.0 \ \dots \ 0.002)}^{16\text{개 요소}}$

8.5.3 단어 임베딩

- 단어 임베딩 공간에서 다층 퍼셉트론으로 IMDB 인식 [프로그램 8-7(b)]
 - 단어 임베딩 기술을 이용하여 IMDB의 샘플을 긍정과 부정으로 분류

프로그램 8-7(b)

단어 임베딩을 사용해 IMDB 데이터를 부정/긍정 평가로 분류하는 학습 모델

```
20
21 embed_space_dim=16 # 16차원의 임베딩 공간
22
23 x_train=preprocessing.sequence.pad_sequences(x_train,maxlen=sample_siz)
24 x_test=preprocessing.sequence.pad_sequences(x_test,maxlen=sample_siz)
25
```

샘플의 크기를 sample_siz(512)로 고정
512보다 작은 샘플은 pad_sequences 함수가 특수 문자로 채움

8.5.3 단어 임베딩

```

26 # 신경망 모델 설계와 학습
27 embed=Sequential()
28 embed.add(Embedding(input_dim=dic_siz,output_dim=embed_space_dim,input_
length=sample_siz))
29 embed.add(Flatten())
30 embed.add(Dense(32,activation='relu'))
31 embed.add(Dense(1,activation='sigmoid'))
32 embed.compile(loss='binary_crossentropy',optimizer='Adam',metrics=
['accuracy'])
33 hist=embed.fit(x_train,y_train,epochs=20,batch_size=64,validation_data=
(x_test,y_test),verbose=2)
34
35 embed.summary()
36
37 # 모델 평가
38 res=embed.evaluate(x_test,y_test,verbose=0)
39 print("정확률은",res[1]*100)
40
41 import matplotlib.pyplot as plt
42
43 # 학습 곡선
44 plt.plot(hist.history['accuracy'])
45 plt.plot(hist.history['val_accuracy'])
46 plt.title('Model accuracy')
47 plt.ylabel('Accuracy')
48 plt.xlabel('Epoch')
49 plt.legend(['Train','Validation'], loc='best')
50 plt.grid()
51 plt.show()

```

Embedding 함수는 input_dim 차원을
output_dim 차원으로 축소

input_length*output_dim 구조의 텐서를 일렬로 펼침

0(부정) 또는 1(긍정)로 분류하므로 출력 노드는 1개
손실 함수는 binary_crossentropy

8.5.3 단어 임베딩

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/20
25000/25000 - 6s - loss: 0.4595 - accuracy: 0.7503 - val_loss: 0.2825 - val_accuracy: 0.8812
...
```

```
Epoch 20/20
25000/25000 - 5s - loss: 1.6844e-05 - accuracy: 1.0000 - val_loss: 0.7613 - val_accuracy: 0.8688
```

Model: "sequential_33"

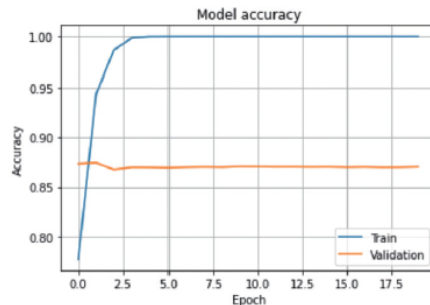
Layer (type)	Output Shape	Param #
embedding_32 (Embedding)	(None, 512, 16)	160000
flatten_32 (Flatten)	(None, 8192)	0
dense_40 (Dense)	(None, 32)	262176
dense_41 (Dense)	(None, 1)	33

Total params: 422,209

Trainable params: 422,209

Non-trainable params: 0

정확률은 86.87999844551086



모델의 구조 Embedding-Flatten-Dense-Dense

테스트 집합에 대한 정확률

8.5.4 LSTM으로 인식: 조기 멈춤 적용

■ LSTM을 사용하는 [프로그램 8-8]

- 시계열 정보를 반영([프로그램 8-7]은 단지 단어 임베딩에 다층 퍼셉트론을 적용하여 시계열 정보를 이용하지 못함)
- 조기 멈춤 적용([프로그램 8-7]의 실행 결과를 보면 세대 1에서 최고 성능을 이룬 후 개선 없음). 조기 멈춤^{early stopping}은 훈련 집합에 대해 덜 수렴했더라도 검증 집합에 대해 성능 개선이 없으면 학습을 마치는 전략

프로그램 8-8

워드 임베딩을 사용하여 IMDB 데이터를 부정/긍정 평가로 분류하는 LSTM 신경망 학습

```

01 from tensorflow.keras.datasets import imdb
02 from tensorflow.keras.models import Sequential
03 from tensorflow.keras.layers import Dense,LSTM,Embedding
04 from tensorflow.keras import preprocessing
05 from tensorflow.keras.callbacks import EarlyStopping
06
07 dic_siz=10000      # 사전의 크기(사전에 있는 단어 개수)
08 sample_siz=512    # 샘플의 크기
09
10 # tensorflow가 제공하는 간소한 버전의 IMDB 읽기
11 (x_train,y_train),(x_test,y_test)=imdb.load_data(num_words=dic_siz)
12
13 embed_space_dim=16 # 16차원의 임베딩 공간
14
15 x_train=preprocessing.sequence.pad_sequences(x_train,maxlen=sample_siz)
16 x_test=preprocessing.sequence.pad_sequences(x_test,maxlen=sample_siz)
17

```

따라해보기

8.5.4 LSTM으로 인식: 조기 멈춤 적용

```

18 early=EarlyStopping(monitor='val_accuracy',patience=5,restore_best_
   weights=True)
19
20 # 신경망 모델의 설계와 학습(LSTM 층 포함)
21 embed=Sequential()
22 embed.add(Embedding(input_dim=dic_siz,output_dim=embed_space_dim,input_
   length=sample_siz))
23 embed.add(LSTM(units=32))
24 embed.add(Dense(1,activation='sigmoid'))
25 embed.compile(loss='binary_crossentropy',optimizer='Adam',metrics=['accuracy'])
26 hist=embed.fit(x_train,y_train,epochs=20,batch_size=64,validation_split=0.2,v
   erbose=2,callbacks=[early])
27
28 # 모델 평가
29 res=embed.evaluate(x_test,y_test,verbose=0)
30 print("정확률은",res[1]*100)
31
32 import matplotlib.pyplot as plt
33
34 # 학습 곡선
35 plt.plot(hist.history['accuracy'])
36 plt.plot(hist.history['val_accuracy'])
37 plt.title('Model accuracy')
38 plt.ylabel('Accuracy')
39 plt.xlabel('Epoch')
40 plt.legend(['Train','Validation'], loc='best')
41 plt.grid()
42 plt.show()

```

가장 높은 성능을 발휘했을 때의 가중치를 취함
5 세대 동안 성능 향상 없으면 멈춤

검증 집합에 대한 정확률을 조기 멈춤 기준으로 사용

콜백 함수를 통해 조기 멈춤 적용

훈련 집합의 20%를 떼어 검증 집합으로 사용

8.5.4 LSTM으로 인식: 조기 멈춤 적용

Train on 20000 samples, validate on 5000 samples

Epoch 1/20
20000/20000 - 119s - loss: 0.4790 - accuracy: 0.7628 - val_loss: 0.3398 - val_accuracy: 0.8640

Epoch 2/20
20000/20000 - 113s - loss: 0.2587 - accuracy: 0.9004 - val_loss: 0.3117 - val_accuracy: 0.8722

Epoch 3/20
20000/20000 - 109s - loss: 0.1976 - accuracy: 0.9288 - val_loss: 0.3347 - val_accuracy: 0.8784

Epoch 4/20
20000/20000 - 119s - loss: 0.1684 - accuracy: 0.9401 - val_loss: 0.3417 - val_accuracy: 0.8754

Epoch 5/20
20000/20000 - 107s - loss: 0.1239 - accuracy: 0.9588 - val_loss: 0.3424 - val_accuracy: 0.8682

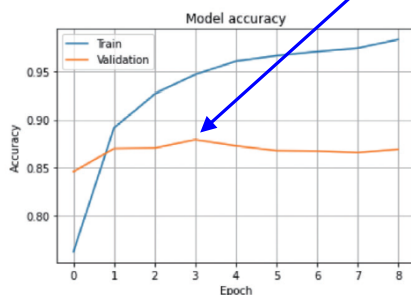
Epoch 6/20
20000/20000 - 105s - loss: 0.0987 - accuracy: 0.9685 - val_loss: 0.4095 - val_accuracy: 0.8734

Epoch 7/20
20000/20000 - 105s - loss: 0.0908 - accuracy: 0.9709 - val_loss: 0.4010 - val_accuracy: 0.8618

Epoch 8/20
20000/20000 - 104s - loss: 0.1116 - accuracy: 0.9596 - val_loss: 0.4849 - val_accuracy: 0.8638

정확률은 86.76400184631348

세대 3 이후 다섯 세대 동안 성능 향상이 없어 학습을 멈춤



8.5.4 LSTM으로 인식: 조기 멈춤 적용

■ [프로그램 8-8]과 [프로그램 8-7]의 성능 분석

- [프로그램 8-8]은 LSTM을 사용해 시계열 특성을 반영하고 조기 멈춤을 적용해 더 유리한 상황인데 [프로그램 8-7]보다 열등. 왜?
 - 두 프로그램 모두 단어의 빈도수에 따라 분류하는 듯
 - boring, terrible, bad 등이 많으면 부정, wonderful, good 등이 자주 나타나면 긍정으로 분류
 - 문장의 의미를 파악하지 못한 채 분류
 - 예, "To me all of the movies are terrible, but this one is not." 문장에서 terrible이 있다는 이유로 두 프로그램 모두 부정으로 분류

■ 문장의 의미 이해하려면 발전된 자연어 처리 알고리즘 필요

- 다음 절에서 word2vec과 GloVe를 간략히 소개

8.5.5 word2vec과 GloVe

■ 단어를 벡터 공간에 표현하는 단어 임베딩 기술

- 오래 전부터 연구되어온 아이디어
- 영국의 언어학자 퍼스 "You shall know a word by the company it keeps." 단어 간의 상호 작용이 매우 중요하다는 통찰
 - 예) "영화"라는 단어는 "아카데미", "흥행" 등의 단어와 함께 등장할 가능성 높음
- 고전적인 기법들: TF(term frequency), LSA, 신경망 기법들
- 2010년대에는 딥러닝을 활용한 단어 임베딩 기술이 주류
 - Word2vec(구글)
 - 1000억개 가량의 뉴스를 모아둔 데이터셋으로 학습. 300만개 가량의 단어를 300차원 공간에 표현
 - GloVe(스탠퍼드 대학)
 - 위키피디아 문서 데이터를 사용하여 학습. 40만개 가량의 단어를 50, 100, 200, 300 차원 공간에 표현

8.5.5 word2vec과 GloVe

■ GloVe로 단어 연관 관계를 찾는 프로그래밍 실습

- 파일 크기가 상대적으로 작은 GloVe를 가지고 실습
- 100차원 파일인 glove.6B.100d.txt 사용

프로그램 8-9(a)

GloVe를 이용해 단어 임베딩 이해하기

```
01 import os
02 import numpy as np
03 from scipy.spatial import distance
04
05 # IMDB 원본 데이터 읽기
06 fname='./glove.6B/glove.6B.100d.txt'
07 f=open(fname,encoding='utf8')
08
09 for line in f:      # 내용을 살필 목적으로 첫 번째 단어만 출력
10     print(type(line))
11     print(line)
12     break
13
```

거리 계산용 함수

100차원 짜리 사용

8.5.5 word2vec과 GloVe

```

14 # 사전 구축(딕셔너리 자료구조로 표현)
15 dictionary={}
16 for line in f:
17     li=line.split()
18     word=li[0]
19     vector=np.asarray(li[1:],dtype='float32')
20     dictionary[word]=vector
21
22 # 가장 가까운 단어를 찾아주는 함수
23 def find_closest_words(vector):
24     return sorted(dictionary.keys(), key=lambda w: distance.euclidean(dictio
        nary[w],vector))
25
26 # 가까운 단어 찾기
27 print(find_closest_words(dictionary['movie'][:5]))
28 print(find_closest_words(dictionary['school'][:5]))
29 print(find_closest_words(dictionary['oak'][:5]))
30
31 # 단어 추론
32 print(find_closest_words(dictionary["seoul"]-dictionary["korea"]+dictionary["
    spain"][:5]))
33 print(find_closest_words(dictionary["animal"]-dictionary["lion"]+dictionary["
    oak"][:5]))
34 print(find_closest_words(dictionary["queen"]-dictionary["king"]+dictionary["a
    ctress"][:5]))

```

문자열 표현을 벡터 형식으로
변환하고 딕셔너리에 저장

매개변수 vector와 가까운 순으로
정렬한 키를 반환하는 함수

'movie'와 가장 가까운 5개 단어 찾기

Seoul-Korea+Spain	→	Madrid
animal-lion+oak	→	tree
queen-king+actress	→	actor

32~34행 단어 연관 관계 추론

8.5.5 word2vec과 GloVe

<class 'str'> 단어 'the'는 100차원 벡터 (-0.038194,-0.24487,...)로 표현

the -0.038194 -0.24487 0.72812 -0.39961 0.083172 0.043953 -0.39141
 0.3344 -0.57545 0.087459 0.28787 -0.06731 0.30906 -0.26384 -0.13231
 -0.20757 0.33395 -0.33848 -0.31743 -0.48336 0.1464 -0.37304 0.34577
 0.052041 0.44946 -0.46971 0.02628 -0.54155 -0.15518 -0.14107 -0.039722
 0.28277 0.14393 0.23464 -0.31021 0.086173 0.20397 0.52624 0.17164
 -0.082378 -0.71787 -0.41531 0.20335 -0.12763 0.41367 0.55187 0.57908
 -0.33477 -0.36559 -0.54857 -0.062892 0.26584 0.30205 0.99775 -0.80481
 -3.0243 0.01254 -0.36942 2.2167 0.72201 -0.24978 0.92136 0.034514
 0.46745 1.1079 -0.19358 -0.074575 0.23353 -0.052062 -0.22044 0.057162
 -0.15806 -0.30798 -0.41625 0.37972 0.15006 -0.53212 -0.2055 -1.2526
 0.071624 0.70565 0.49744 -0.42063 0.26148 -1.538 -0.30223 -0.073438
 -0.28312 0.37104 -0.25217 0.016215 -0.017099 -0.38984 0.87424 -0.72569
 -0.51058 -0.52028 -0.1459 0.8278 0.27062

['movie', 'film', 'movies', 'films', 'hollywood']

['school', 'college', 'schools', 'elementary', 'students'] 가까운 단어 5개 찾기 결과

['oak', 'pine', 'cedar', 'walnut', 'grove']

['madrid', 'spain', 'santiago', 'seville', 'valencia']

['oak', 'trees', 'woodland', 'wood', 'organic']

단어 연관 관계 찾기 결과

['actress', 'actresses', 'dancer', 'actor', 'comedienne']

8.5.5 word2vec과 GloVe

- 고차원을 2차원 또는 3차원으로 축소하여 데이터를 가시화하는 기법
 - PCA, iso-contour 등 많은 기법이 있음
 - tsne가 가장 뛰어남
- [프로그램 8-9(b)]는 tsne를 사용하여 100차원 벡터를 2차원으로 변환

프로그램 8-9(b)

tsne를 이용한 시각화

```
35
36 from sklearn.manifold import TSNE
37 import matplotlib.pyplot as plt
38
39 # tsne를 이용하여 2차원 공간으로 축소하고 시각화
40 tsne=TSNE(n_components=2,random_state=0)
41 words=list(dictionary.keys())
42 vectors=[dictionary[word] for word in words]
43 p2=tsne.fit_transform(vectors[:100])
44 plt.scatter(p2[:,0],p2[:,1])
45
46 for label,x,y in zip(words,p2[:,0],p2[:,1]):
47     plt.annotate(label,xy=(x,y))
```

2차원으로 축소하라는 지시

앞에 있는 100개만 시각화

8.5.5 word2vec과 GloVe

