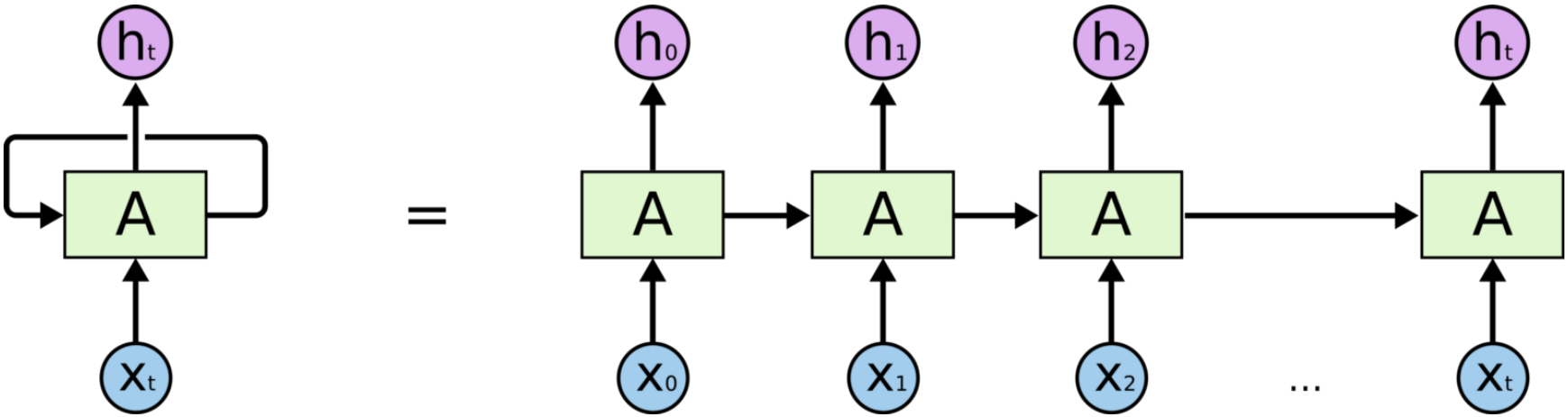
순환 신경망(RNN)

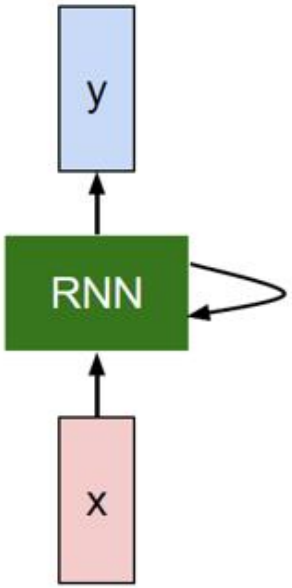
2019.12

■ 순환 신경망(RNN: Recurrent Neural Network)

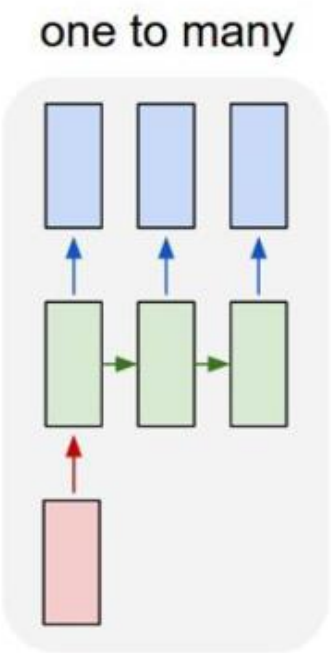
- Sequence Data
  - 문장(Text), 음성 신호, 주가 데이터
- DNN, CNN: 입력층 → 출력층 한 방향으로만 흐르는 Feed forward 신경망
- RNN
  - 시퀀스 데이터의 모델링에 사용되는 신경망



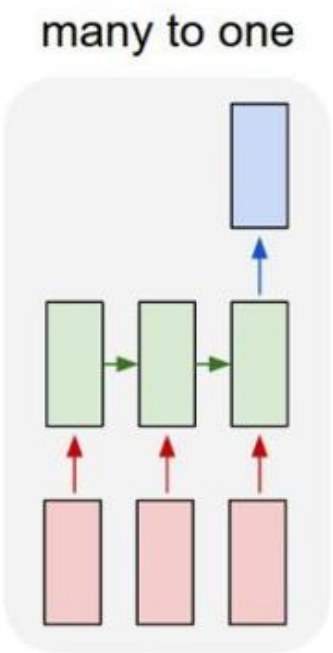
■ 입력/출력 시퀀스



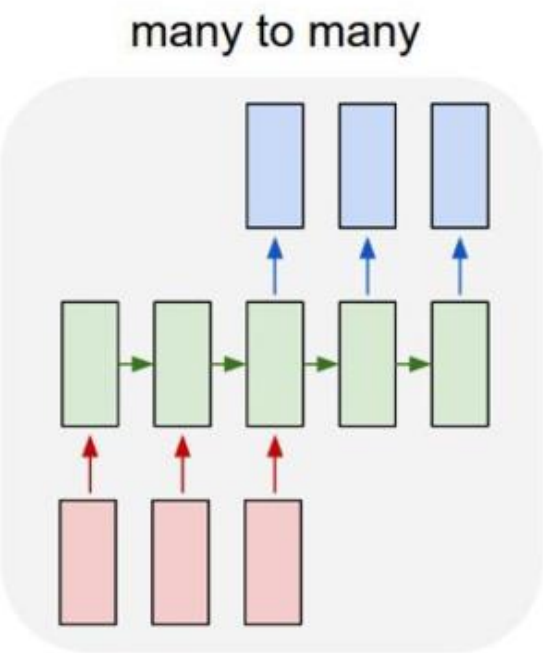
<기본구조>



<사진설명 붙이기>
사진 → 단어들

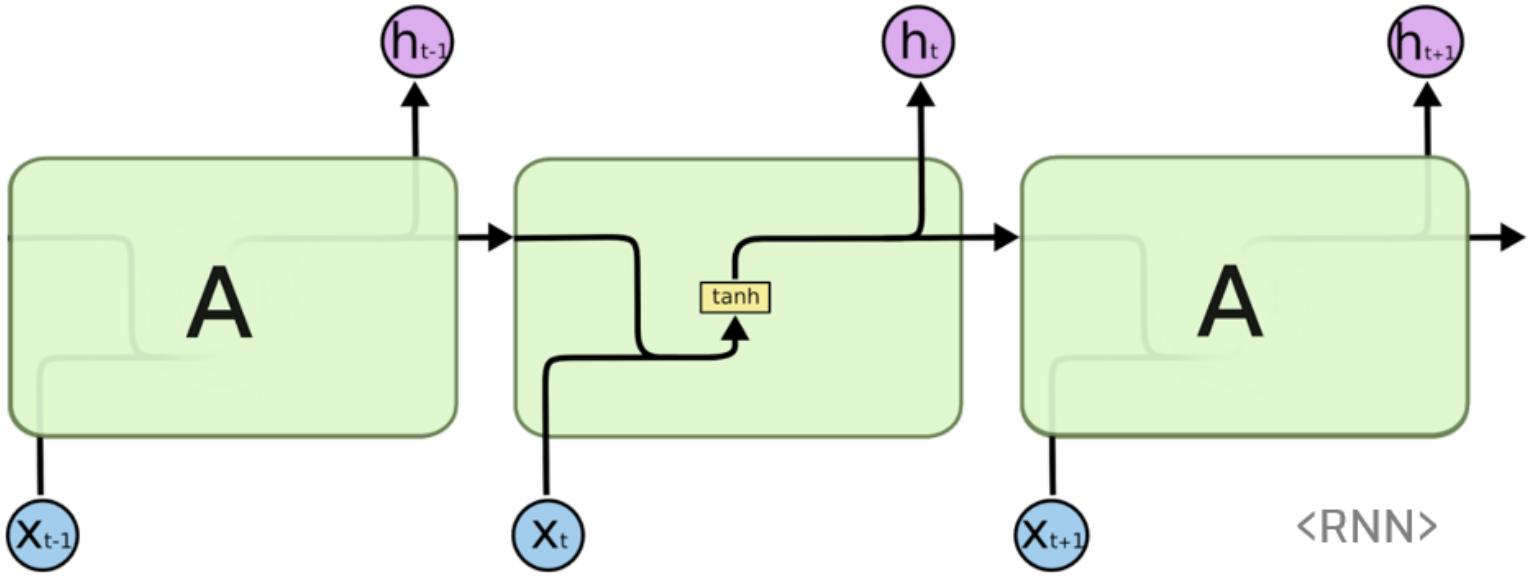


<감성분석>
단어들 → 감성점수



<번역>
단어들 → 단어들

■ 기본적인 구조(Simple RNN, Vanilla RNN)



$$\boldsymbol{h_t = \tanh(W_x x_t + W_h h_{t-1} + b)}$$

활성화 함수 가중치 바이어스

■ Simple RNN code (One-to-many 구조)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN

model = Sequential()
model.add(SimpleRNN(3, input_shape=(2,10)))
# model.add(SimpleRNN(hidden_size=3, input_length=2, input_dim=10))
model.summary()
```

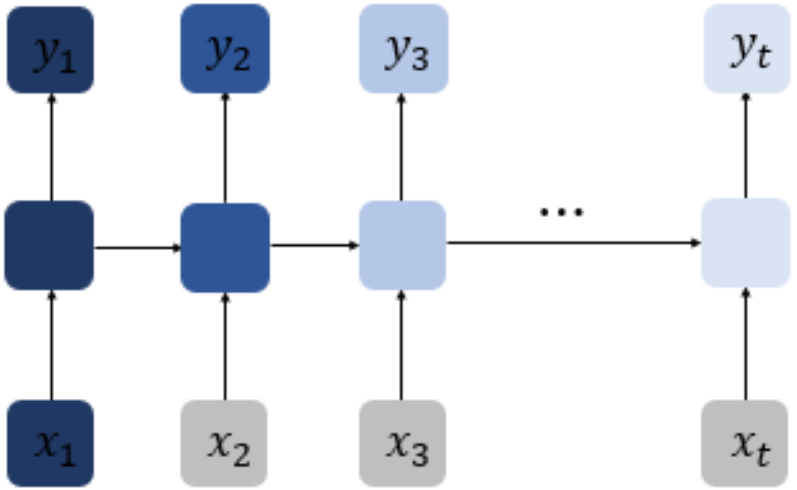
▶ 01_Text_Generation-RNN.ipynb

■ 장단기 메모리(Long Short-Term Memory, LSTM)

❖ 바닐라 RNN의 한계

- 비교적 짧은 시퀀스에 대해서만 효과를 보임
- 시점(time step)이 길어질수록 앞의 정보가 뒤로 충분히 전달되지 못함

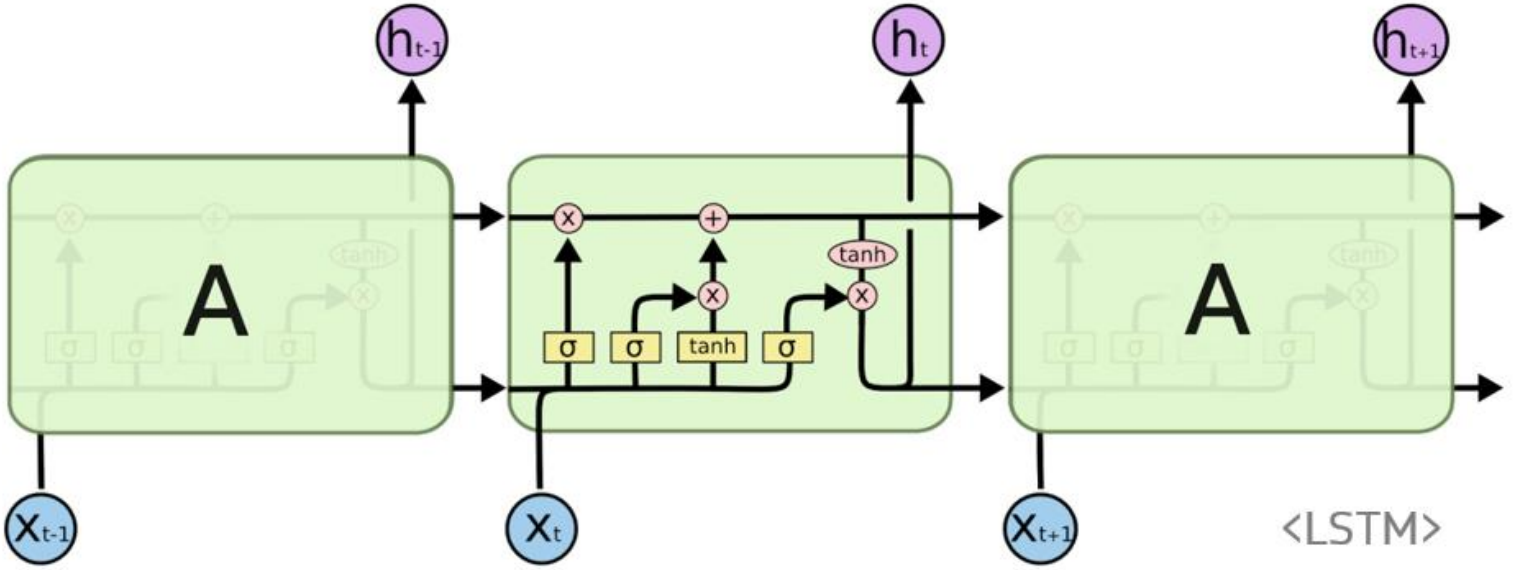
  ➔ 장기 의존성 문제(Problem of Long-Term Dependencies)



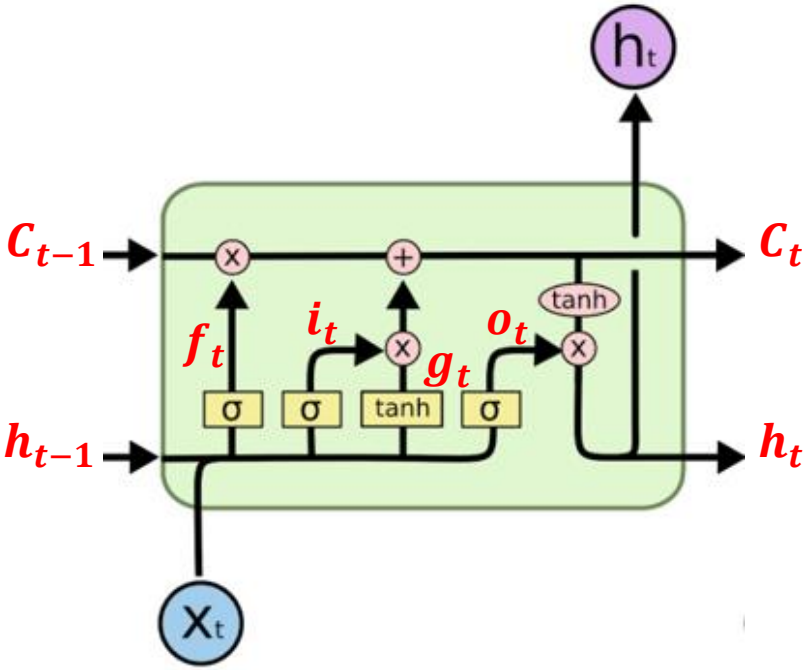
"모스크바에 여행을 왔는데 건물도 예쁘고 먹을 것도 맛있었어. 그런데 글쎄 직장 상사한테 전화가 왔어. 어디냐고 묻더라구 그래서 나는 말했지. 저 여행왔는데요. 여기 ____"

■ 장단기 메모리(Long Short-Term Memory, LSTM)

- 은닉층의 메모리 셀에 입력 게이트, 망각(삭제) 게이트, 출력 게이트를 추가
- 불필요한 기억은 지우고, 기억해야 할 것들은 기억함
- LSTM은 한 층 안에서 반복을 많이 해야 하는 RNN의 특성상 일반 신경망보다 기울기 소실 문제가 더 많이 발생하고 이를 해결하기 어렵다는 단점을 보완한 방법
- 즉, 반복되기 직전에 다음 층으로 기억된 값을 넘길지 안 넘길지를 관리하는 단계를 하나 더 추가하는 것



■ 장단기 메모리(Long Short-Term Memory, LSTM)



- 입력 게이트 :

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

- 삭제 게이트 :

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

- 셀 상태(장기 상태) :

$$C_t = f_t \circ C_{t-1} + i_t \circ g_t$$

- 출력 게이트와 은닉 상태(단기 상태) :

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$h_t = o_t \circ \tanh(C_t)$$

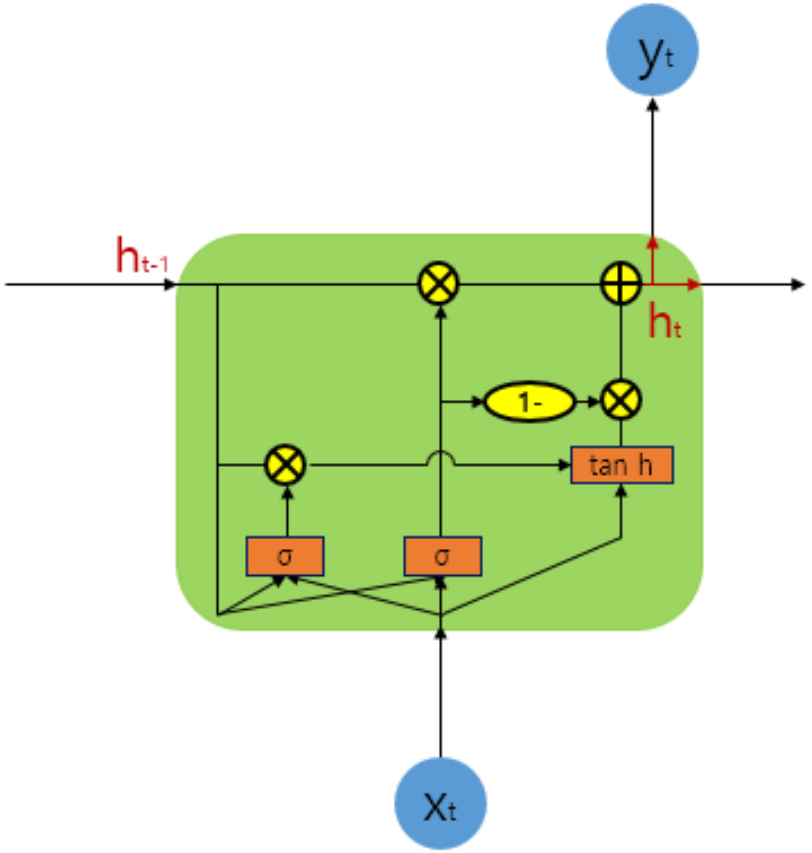
■ LSTM code (One-to-many 구조)

▶ 02_Text_Generation-LSTM.ipynb

■ 게이트 순환 유닛(Gated Recurrent Unit, GRU)

- LSTM의 장기 의존성 문제에 대한 해결책을 유지하면서, 은닉 상태를 업데이트하는 계산을 줄인 RNN
- 업데이트 게이트와 리셋 게이트 두 가지 게이트만이 존재
- LSTM보다 학습 속도가 빠르다고 알려져 있지만 여러 평가에서 GRU는 LSTM과 비슷한 성능을 보인다고 알려져 있음
- 사용 방법은 SimpleRNN이나 LSTM과 동일

```
GRU(hidden_size,
    input_shape=(timesteps, input_dim))
```

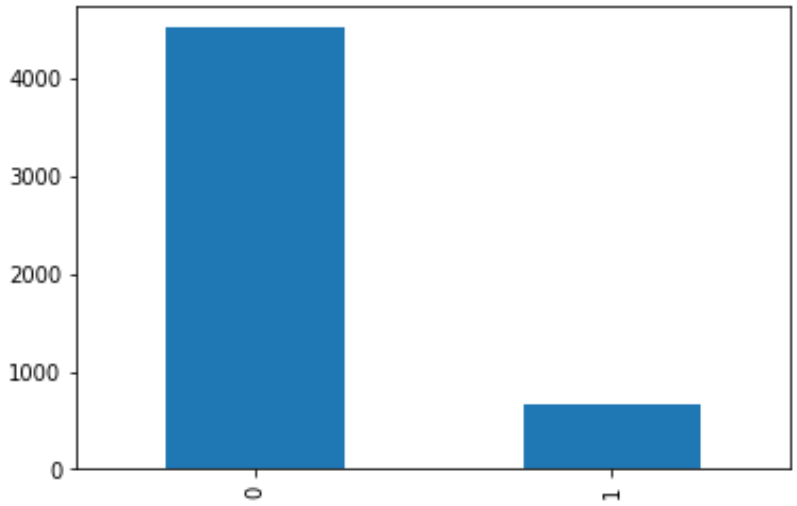


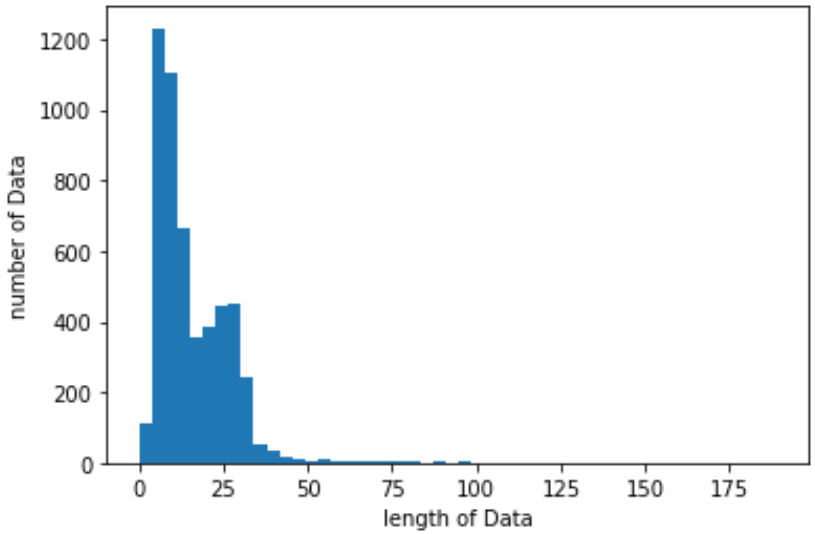
■ 텍스트 분류 개요

- 지도 학습
- RNN의 다-대-일(Many-to-One) 문제
- 모든 시점(time step)에 대해서 입력을 받지만 최종 시점의 RNN 셀만이 은닉 상태를 출력하고, 이것이 출력층으로 가서 활성화 함수를 통해 정답을 고르는 문제
- 이진 분류
  - sigmoid 함수, binary_crossentropy, 출력층 크기 = 1
  - 스팸 메일 분류하기, IMDB 리뷰 감성 분류하기
- 다중 클래스 분류
  - softmax 함수, categorical_crossentropy, 출력층 크기 = N
  - 로이터 뉴스 분류하기

■ 스팸 메일 분류

- 캐글에서 제공하는 스팸메일 데이터 활용
  - 총 5,572개의 데이터, 중복 데이터 403개
  - 중복 제거한 5,169개의 데이터 중 햄 4,516개, 스팸 653개
  - Unique 단어의 개수 : 8,920 개
  - 메일의 최대 단어 수 : 189 (평균 15.61)
  - 훈련 데이터 80%, 테스트 데이터 20%로 분리





■ 스팸 메일 분류

- 워드 임베딩

	One-hot vector	Embedding vector
차원	고차원(단어 집합의 크기)	저차원
다른 표현	희소 벡터의 일종	밀집 벡터의 일종
값의 지정	수동	훈련 데이터로부터 학습함
값의 타입	1과 0	실수

- 케라스의 Embedding()
  - 단어를 밀집 벡터로 변환한 수
  - 인공신경망 학습과 같이 단어 벡터를 학습하는 방법 사용
  - `Embedding(vocab_size, output_dim, input_length)`

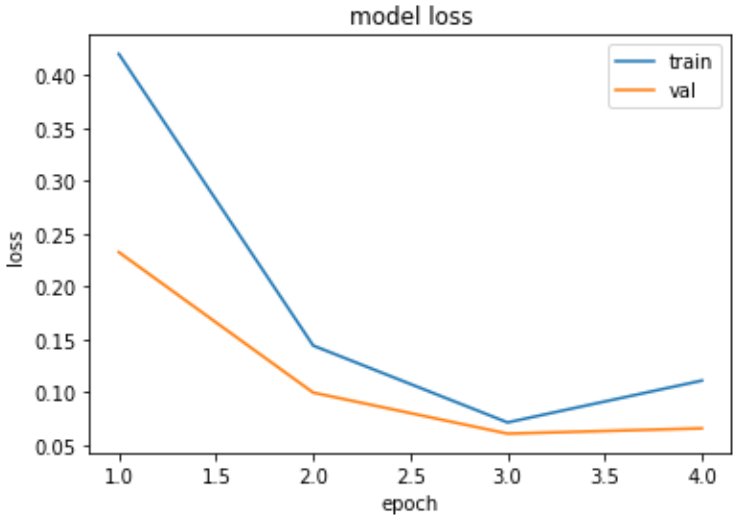
    `입력: (number of samples, input_length)`

■ 스팸 메일 분류

- 모델 설계

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 32)          285472
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 32)                2080
_________________________________________________________________
dense (Dense)                (None, 1)                 33
=================================================================
```

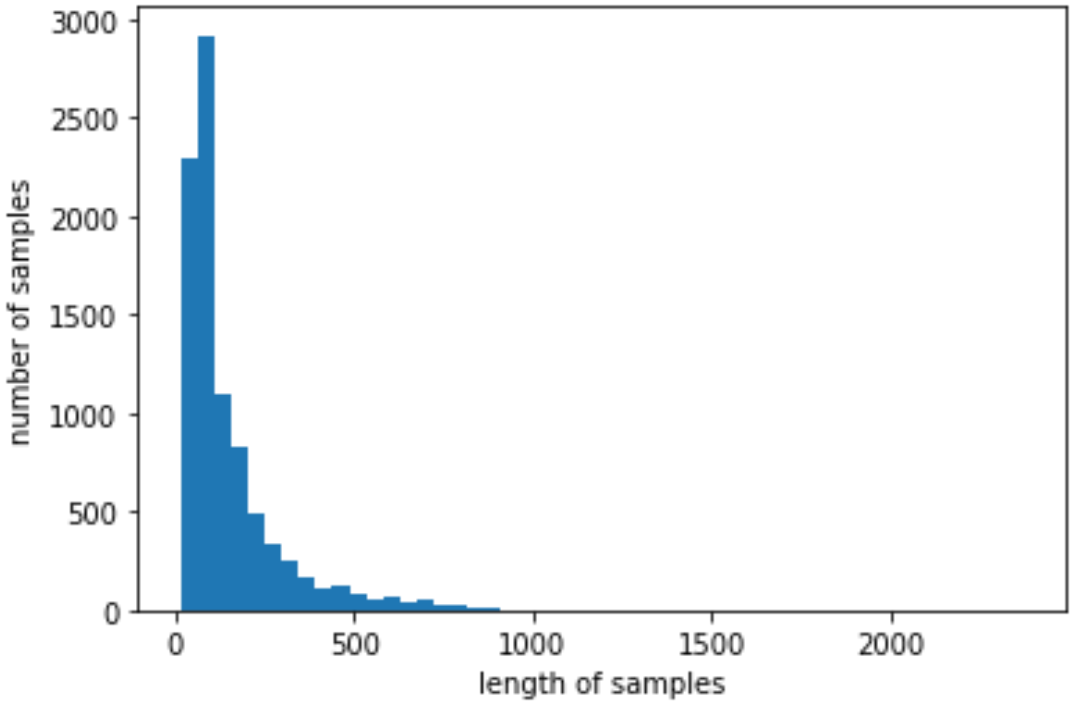
- optimizer='rmsprop', loss='binary_crossentropy'
- 훈련용 데이터 중 20%는 검증용으로 사용
- 테스트 정확도는 98.16%



▶ 11_Spam Mail 분류-RNN.ipynb

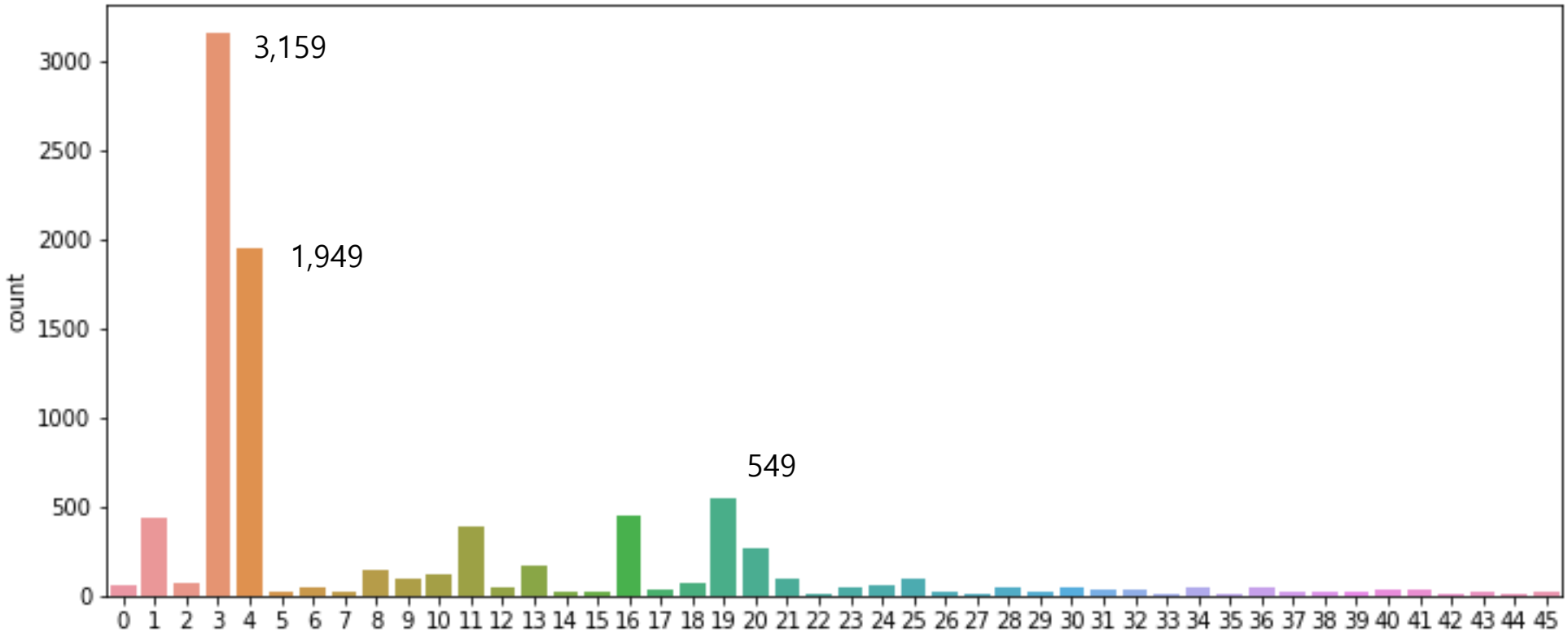
■ 로이터 뉴스 분류

- Keras에서 제공하는 데이터 활용 (tensorflow.keras.datasets.reuters)
- 46개 카테고리의 총 11,228개의 뉴스 기사 데이터
- 전처리가 된 상태로 제공
- 뉴스 기사의 길이
  - 평균: 145.5 개의 단어
  - 최대: 2,376 단어



■ 로이터 뉴스 분류

- 뉴스의 카테고리별 분포



■ 로이터 뉴스 분류

- 단어 분포
  - 고유 단어수: 30,979개
  - 빈도 순으로 index 부여
  - get_word_index() 메소드 제공
  - 빈도수 1위: 'the'
- 첫번째 뉴스

X_train[0]	y_train[0]
[1, 27595, 28842, 8, 43, 10, 447, 5, 25, 207, 270, 5, 3095, 111, 16, 369, 186, 90, 67, 7, 89, 5, 19, 102, 6, 19, 124, 15, 90, 67, 84, 22, 482, 26, 7, 48, 4, 49, 8, 864, 39, 209, 154, 6, 151, 6, 83, 11, 15, 22, 155, 11, 15, 7, 48, 9, 4579, 1005, 504, 6, 258, 6, 272, 11, 15, 22, 134, 44, 11, 15, 16, 8, 197, 1245, 90, 67, 52, 29, 209, 30, 32, 132, 6, 109, 15, 17, 12]	3
the wattie nondiscriminatory mln loss for plc said at only ended said commonwealth could 1 traders now april 0 a after said from 1985 and from foreign 000 april 0 prices its account year a but in this mln home an states earlier and rise and revs vs 000 its 16 vs 000 a but 3 psbr oils several and shareholders and dividend vs 000 its all 4 vs 000 1 mln agreed largely april 0 are 2 states will billion total and against 000 pct dlrs	

■ 로이터 뉴스 분류

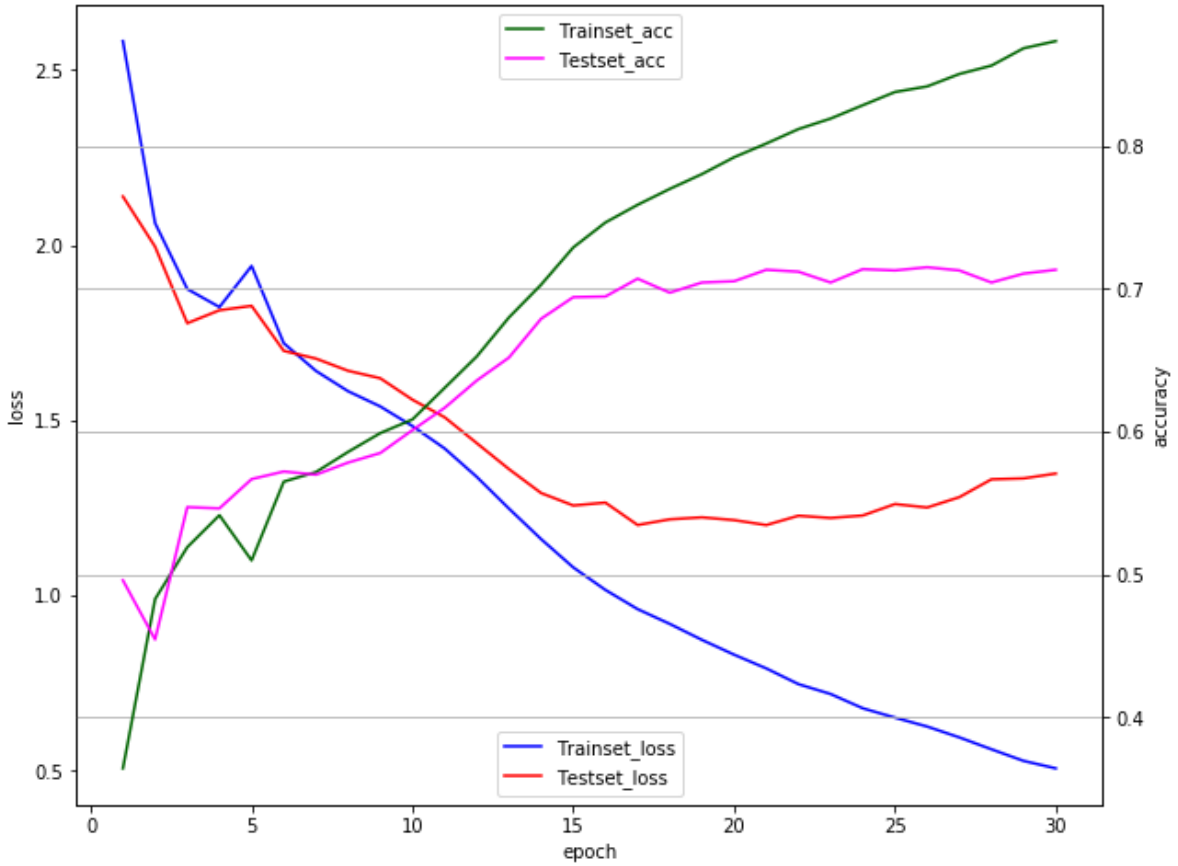
- LSTM 모델
  - 빈도수 1000 까지의 단어만 사용
  - 모든 문장이 아니라 100 단어 까지만 사용
- 120 차원의 Embedding vector

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 120)         120000
_________________________________________________________________
lstm (LSTM)                  (None, 120)               115680
_________________________________________________________________
dense (Dense)                (None, 46)                5566
=================================================================
```

- optimizer='adam', loss='categorical_crossentropy'
- 훈련용 데이터 중 20%는 검증용으로 사용

■ 로이터 뉴스 분류

- 정확도 : 71.33%

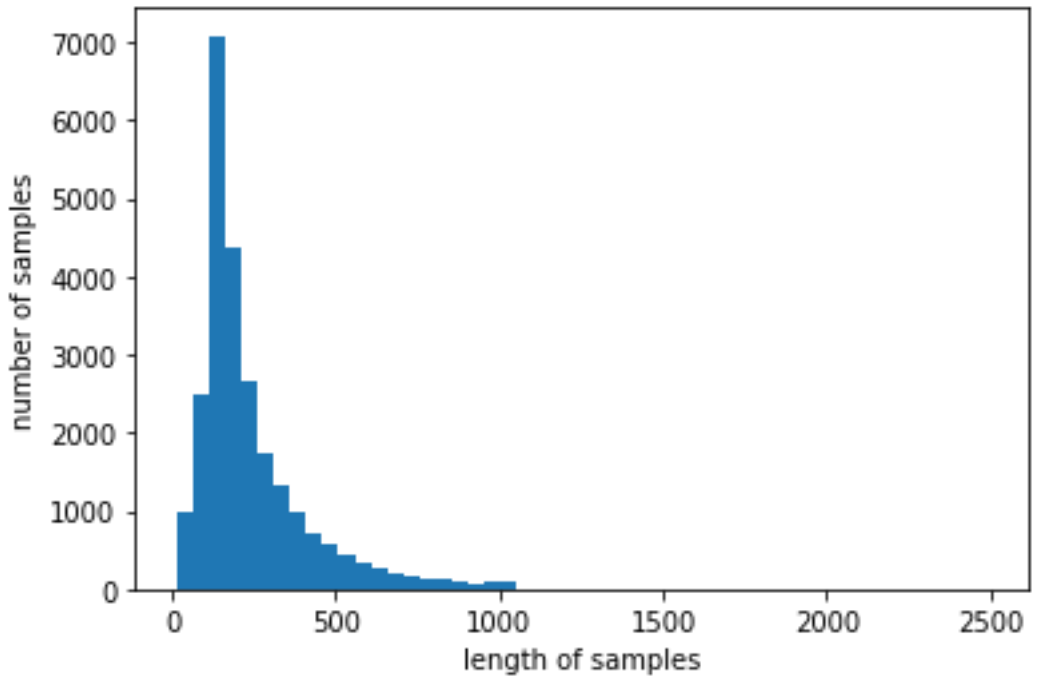


▶ 12_로이터 뉴스 카테고리 분류-LSTM.ipynb

■ IMDB 영화 리뷰 감성 분류

- Keras에서 제공하는 데이터 활용 (tensorflow.keras.datasets.imdb)
- 2개 카테고리(긍정: 1, 부정: 0)의 총 50,000개의 영화 리뷰 데이터
- 스탠포드 대학교에서 2011년에 낸 논문에서 이 데이터를 소개

  훈련 데이터와 테스트 데이터를 50:50대 비율로 분할하여 88.89%의 정확도
- 길이
  - 평균: 238.7 개의 단어
  - 최대: 2,494 단어



■ IMDB 영화 리뷰 감성 분류

- 단어 분포
  - 고유 단어수: 88,584개
- 일곱번째 리뷰

X_train[6]	y_train[6]
[1, 6740, 365, 1234, 5, 1156, 354, 11, 14, 5327, 6638, 7, 1016, 10626, 5940, 356, 44, 4, 1349, 500, 746, 5, 200, 4, 4132, 11, 16393, 9363, 1117, 1831, 7485, 5, 4831, 26, 6, 71690, 4183, 17, 369, 37, 215, 1345, 143, 32677, 5, 1838, 8, 1974, 15, 36, 119, 257, 85, 52, 486, 9, 6, 26441, 8564, 63, 271, 6, 196, 96, 949, 4121, 4, 74170, 7, 4, 2212, 2436, 819, 63, 47, 77, 7175, 180, 6, 227, 11, 94, 2494, 33740, 13, 423, 4, 168, 7, 4, 22, 5, 89, 665, 71, 270, 56, 5, 13, 197, 12, 161, 5390, 99, 76, 23, 77842, 7, 419, 665, 40, 91, 85, 108, 7, 4, 2084, 5, 4773, 81, 55, 52, 1901]	1
the boiled full involving to impressive boring this as murdering naschy br villain council suggestion need has of costumes b message to may of props this echoed concentrates concept issue skeptical to god's he is dedications unfolds movie women like isn't surely i'm rocketed to toward in here's for from did having because very quality it is captain's starship really book is both too worked carl of mayfair br of reviewer closer figure really there will originals things is far this make mistakes kevin's was couldn't of few br of you to don't female than place she to was between that nothing dose movies get are 498 br yes female just its because many br of overly to descent people time very bland	positive

■ IMDB 영화 리뷰 감성 분류

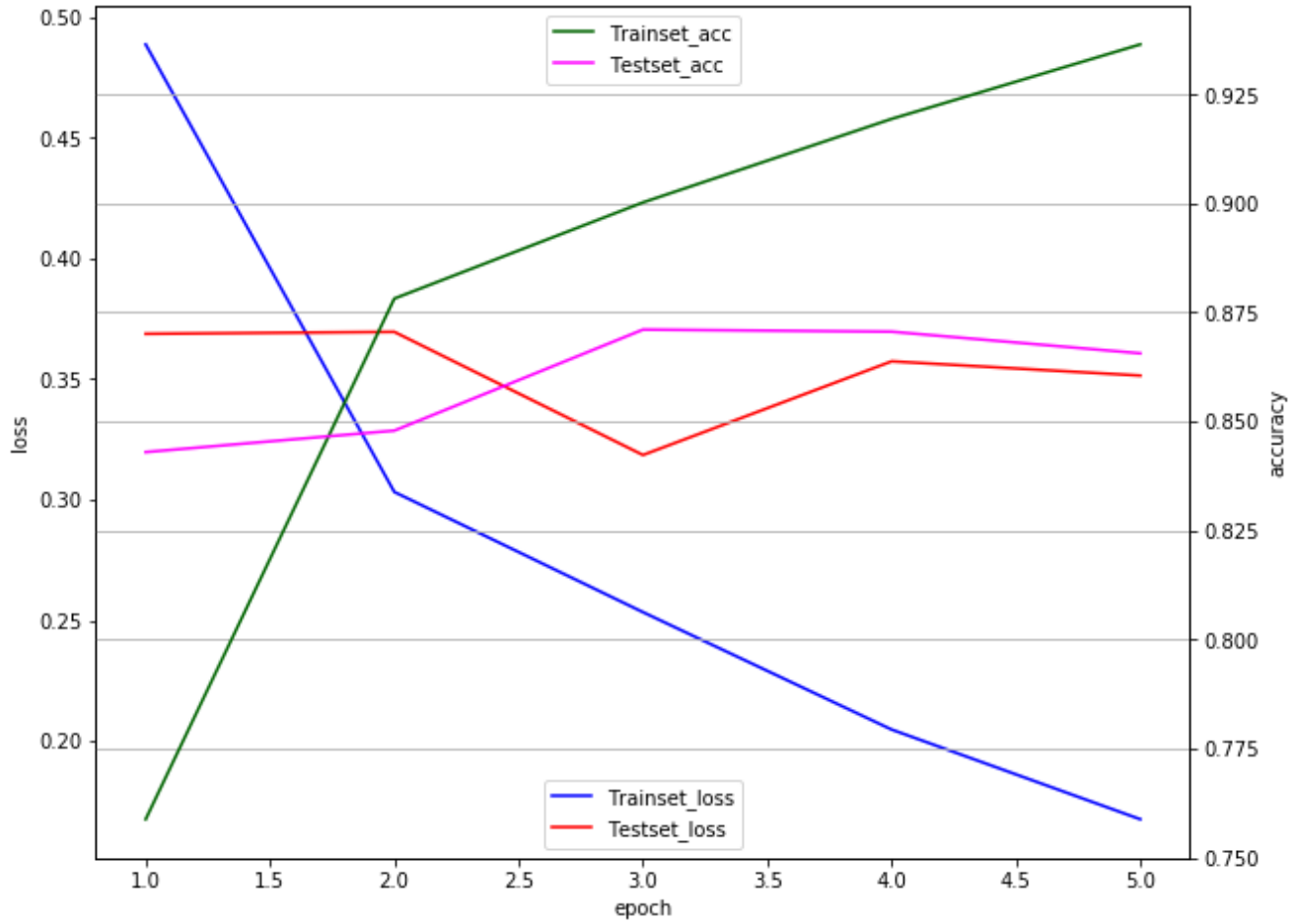
- LSTM 모델
  - 빈도수 5000 까지의 단어만 사용
  - 모든 문장이 아니라 500 단어 까지만 사용
- 120 차원의 Embedding vector

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 120)         600000
_________________________________________________________________
lstm (LSTM)                  (None, 120)               115680
_________________________________________________________________
dense (Dense)                (None, 1)                 121
=================================================================
```

- optimizer='adam', loss='binary_crossentropy'

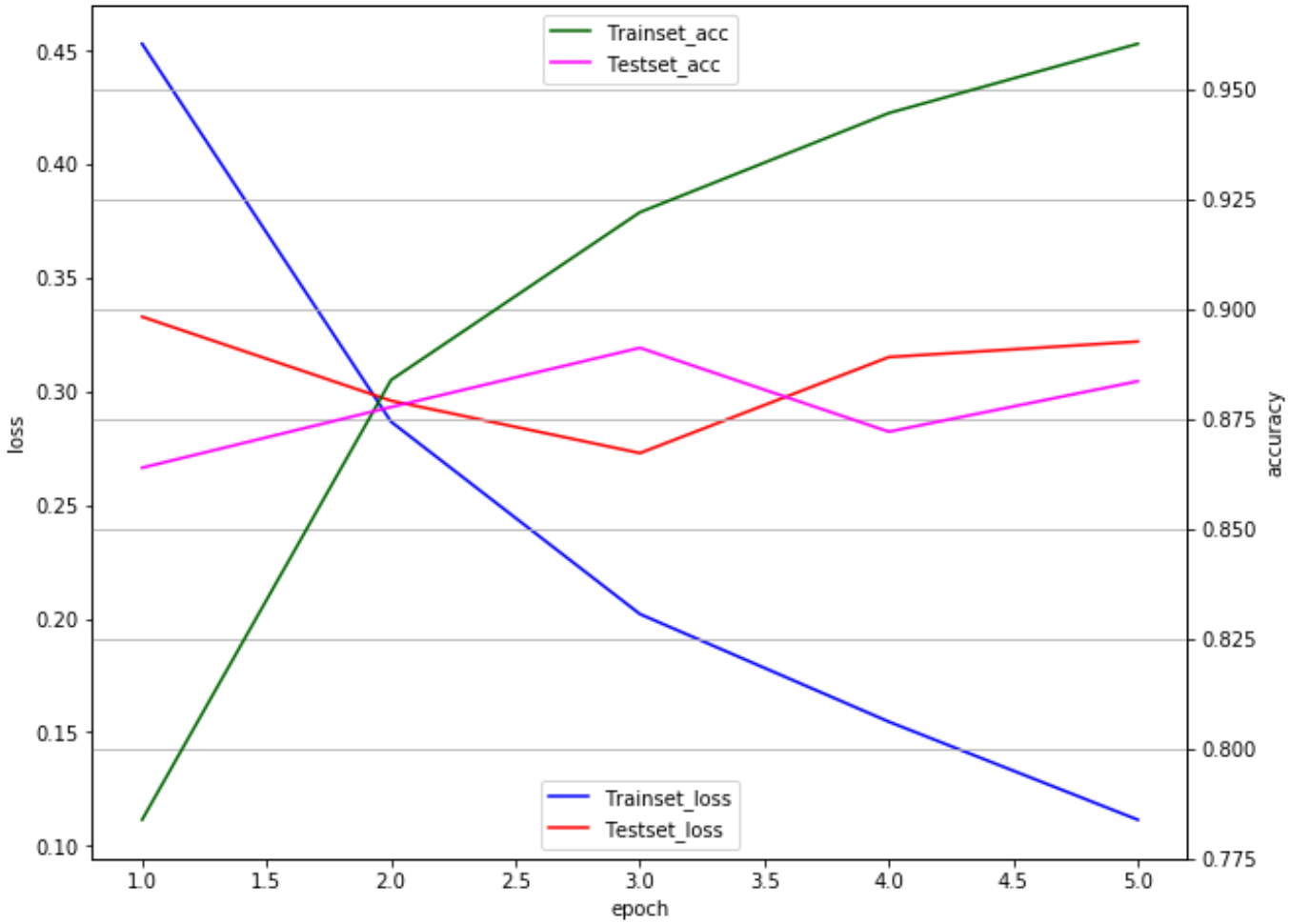
■ IMDB 영화 리뷰 감성 분류

▶ 13_IMDB 영화리뷰 감성 분류-LSTM.ipynb



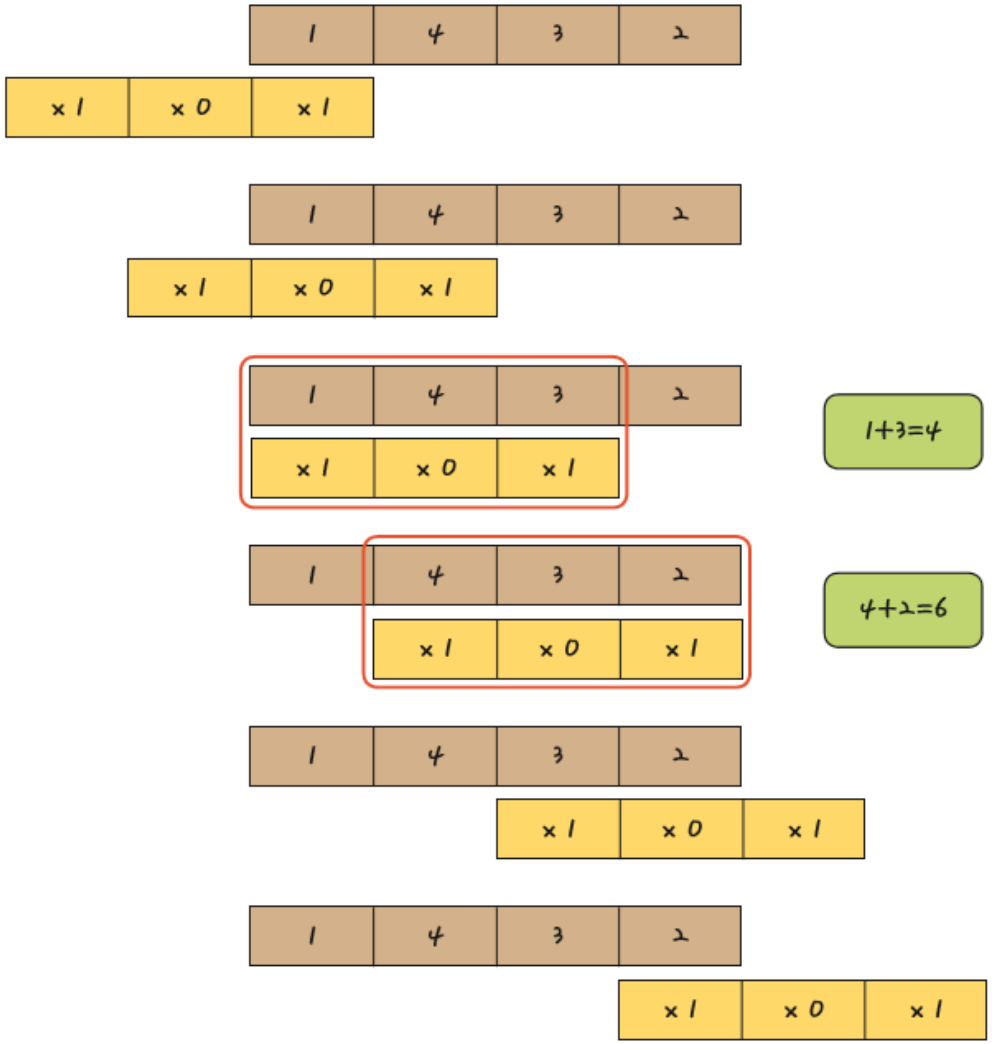
■ IMDB 영화 리뷰 감성 분류

▶ 14_IMDB 영화리뷰 감성 분류-GRU.ipynb



■ IMDB 영화 리뷰 감성 분류 – LSTM + CNN

- Conv1D



■ IMDB 영화 리뷰 감성 분류 – LSTM + CNN

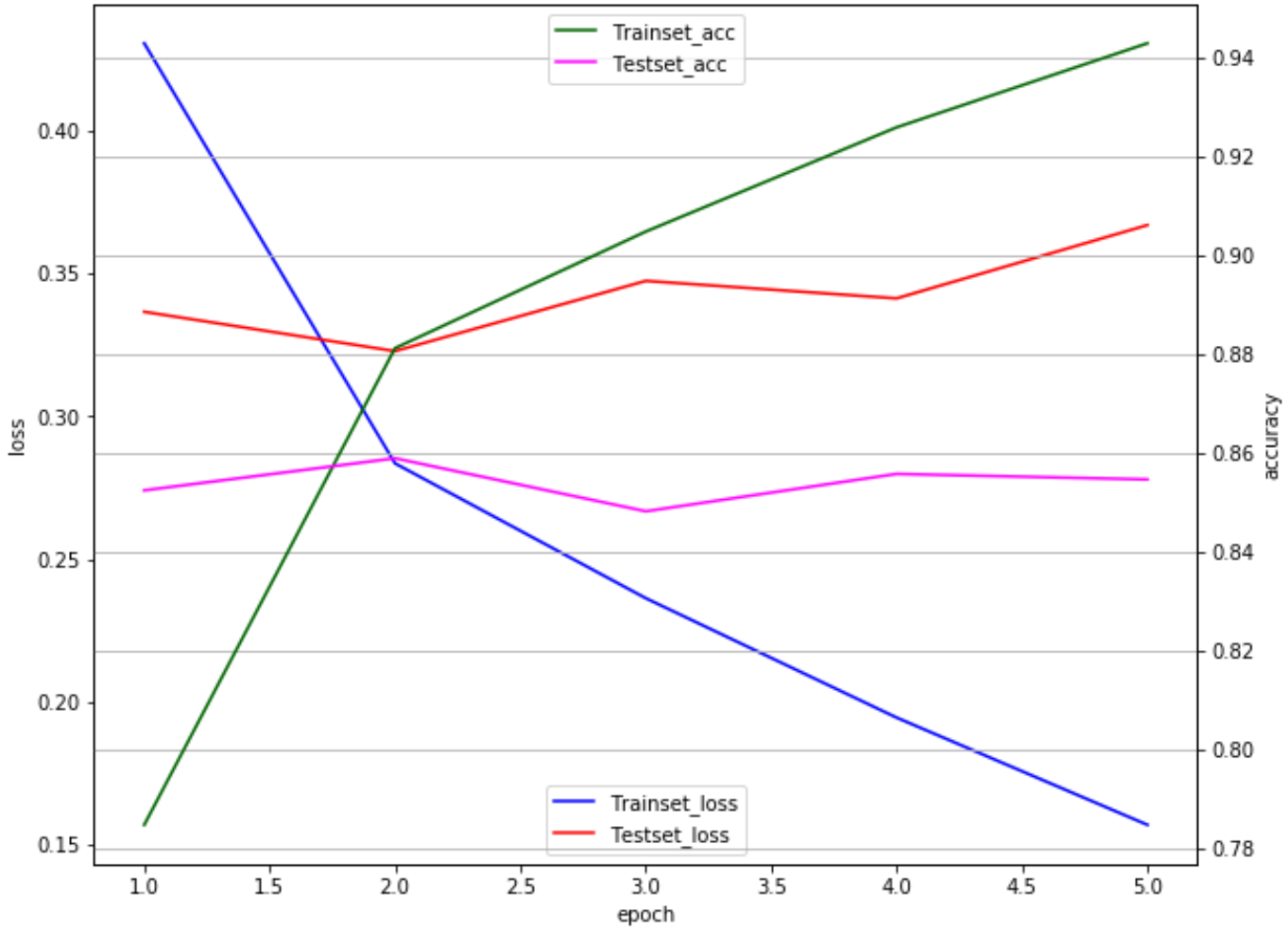
- MaxPooling1D

- Model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 120)         600000
_________________________________________________________________
dropout (Dropout)            (None, None, 120)         0
_________________________________________________________________
conv1d (Conv1D)              (None, None, 64)          38464
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, None, 64)          0
_________________________________________________________________
lstm (LSTM)                  (None, 55)                26400
_________________________________________________________________
dense (Dense)                (None, 1)                 56
=================================================================
```

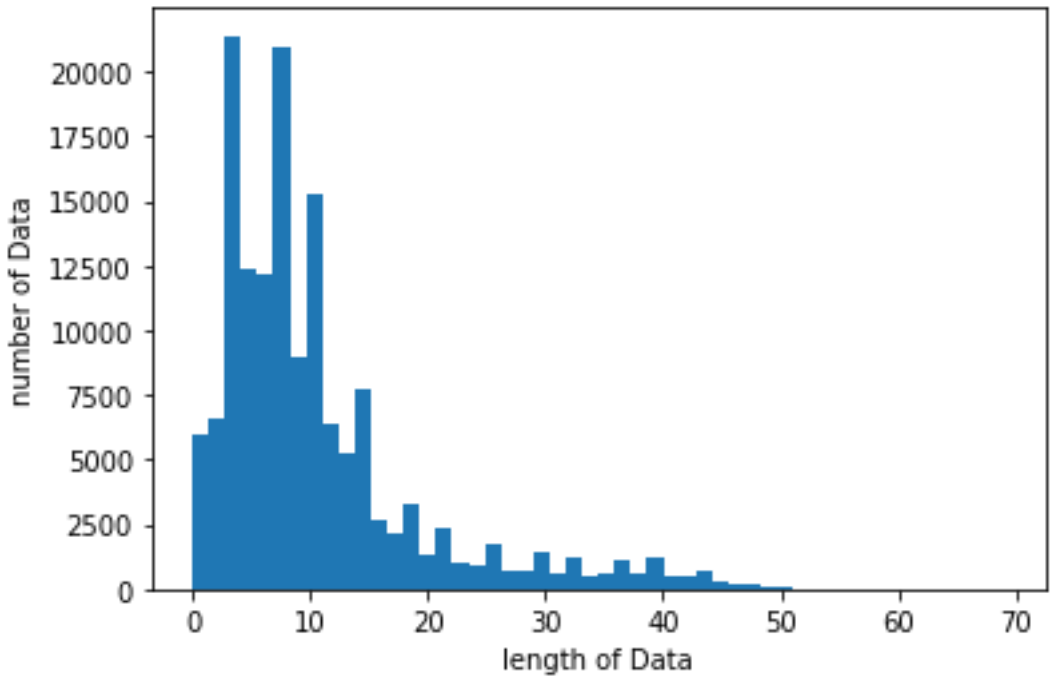
■ IMDB 영화 리뷰 감성 분류 – LSTM + CNN

▶ 15_IMDB 영화리뷰 감성 분류-LSTM-CNN.ipynb



■ 네이버 영화 리뷰 감성 분류

- Github에 올라가 있는 데이터 활용 (https://github.com/e9t/nsmc/)
- 2개 카테고리(긍정: 1, 부정: 0)의 총 200,000개의 영화 리뷰 데이터
  리뷰 점수: 10~9 → 긍정, 4~1 → 부정, 8~5 → 미사용
- 학습용: 150,000개, 테스트용: 50,000개
- 길이
  - 평균: 10.65 개의 단어
  - 최대: 69 단어



■ 네이버 영화 리뷰 감성 분류

❖ 한글 데이터 전처리

- 특수 문자, 영어 등을 모두 제거한 후 한글만 남김 ← 정규 표현식 활용

```
train_data['document'] = train_data['document'].str.replace("[^ㄱ-ㅎㅏ-ㅣ가-힣 ]","")
```

- 토큰화
- 불용어 제거

❖ 모델

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 100)         3500000
_________________________________________________________________
lstm (LSTM)                  (None, 128)               117248
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
```

▶ 16_네이버 영화리뷰 감성 분석-LSTM.ipynb

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 조건부 확률

- A 사건이 발생했을 때 B 사건이 일어날 확률

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- *또는,* B 사건이 발생했을 때 A 사건이 일어날 확률

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

- A, B 사건이 독립인 경우에는 아래 식을 만족한다.

$$P(A \cap B) = P(A)P(B)$$

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 베이즈의 정리

- 용어 정리
  - 사전 확률 : 이미 알고 있는 사건(들)의 확률
  - 우도(Likelihood Probability)

    이미 알고 있는 사건(들)이 발생했다는 조건하에 다른 사건이 발생할 확률
  - 사후 확률 : 사전 확률과 우도 확률을 통해서 알게되는 조건부 확률

- 베이즈 정리(Bayes Theorem)

우도 사전확률

사후확률

$$P(A_k|B) = \frac{P(B|A_k)P(A_k)}{P(B)}$$

주변우도(Marginal Likelihood)

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 나이브 베이즈

- 나이브(Naïve)의 의미
  - '순진한', '순수한' 이라는 뜻
  - 수학에서 단순성을 부여할 때 사용
  - 다양한 세부 요인의 영향력을 모두 동등하고 독립적이라고 가정

❖ 나이브 베이즈 분류기의 예 – 스팸 메일 필터

- 입력 텍스트(메일의 본문)가 주어졌을 때

  P(정상 메일 | 입력 테스트) = 입력 텍스트가 있을 때 정상 메일일 확률

  P(스팸 메일 | 입력 텍스트) = 입력 텍스트가 있을 때 스팸 메일일 확률

- 베이즈 정리에 따라 식을 표현하면

  $P(\text{정상 메일} \mid \text{입력 텍스트}) = P(\text{입력 텍스트} \mid \text{정상 메일}) \times P(\text{정상 메일}) / P(\text{입력 텍스트})$

  $P(\text{스팸 메일} \mid \text{입력 텍스트}) = P(\text{입력 텍스트} \mid \text{스팸 메일}) \times P(\text{스팸 메일}) / P(\text{입력 텍스트})$

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

- 입력 텍스트가 주어졌을 때, P(정상 메일 | 입력 텍스트)가 P(스팸 메일 | 입력 텍스트) 보다 크면 정상 메일

  P(정상 메일 | 입력 텍스트) = P(입력 텍스트 | 정상 메일) × P(정상 메일)

  P(스팸 메일 | 입력 텍스트) = P(입력 텍스트 | 스팸 메일) × P(스팸 메일)

- 입력 텍스트는 메일의 본문

  메일 본문에 있는 모든 단어를 토큰화시켜서 이 단어들을 나이브 베이즈 분류기의 입력으로 사용

- 만약 메일 본문에 있는 단어가 3개라고 가정($w1$, $w2$, $w3$)

  나이브 베이즈 분류기는 모든 단어가 독립적이라고 가정

  P(정상 메일 | 입력 텍스트) = P($w1$ | 정상 메일) × P($w2$ | 정상 메일) × P($w3$ | 정상 메일) × P(정상 메일)

  P(스팸 메일 | 입력 텍스트) = P($w1$ | 스팸 메일) × P($w2$ | 스팸 메일) × P($w3$ | 스팸 메일) × P(스팸 메일)

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 스팸 메일 분류기

- 훈련 데이터

	메일로부터 토큰화 및 정제된 단어	분류
1	me free lottery	spam
2	free get free you	spam
3	you free scholarship	ham
4	free to contact me	ham
5	you won award	ham
6	you ticket lottery	spam

- 'you free lottery' 는 spam인가 ham인가?

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 스팸 메일 분류기

- 'you free lottery' 에 대해 정상 메일일 확률과 스팸 메일일 확률

  P(정상 메일 | 입력 텍스트) = P(you | 정상 메일) × P(free | 정상 메일)
  × P(lottery | 정상 메일) × P(정상 메일)

  P(스팸 메일 | 입력 텍스트) = P(you | 스팸 메일) × P(free | 스팸 메일)
  × P(lottery | 스팸 메일) × P(스팸 메일)

  P(정상 메일) = P(스팸 메일) = 총 메일 6개 중 3개 = 0.5

- P(you | 정상 메일)를 구하는 방법

  분모: 정상 메일에 등장한 모든 단어의 빈도 수의 총합

  분자: 정상 메일에서 you가 총 등장한 빈도 수

- 확률 계산

  P(정상 메일 | 입력 텍스트) = 2/10 × 2/10 × 0/10 = 0

  P(스팸 메일 | 입력 텍스트) = 2/10 × 3/10 × 2/10 = 0.012

- P(정상 메일 | 입력 텍스트) < P(스팸 메일 | 입력 텍스트)이므로 스팸 메일

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 장점

- 간단하고, 빠르며, 정확한 모델
- computation cost가 작음 (따라서 빠름)
- 큰 데이터셋에 적합
- 연속형보다 이산형 데이터에서 성능이 좋음
- Multiple class 예측을 위해서도 사용 가능

❖ 단점

- feature 간의 독립성이 있어야 함
  하지만 실제 데이터에서 모든 feature가 독립인 경우는 희박함

■ 나이브 베이즈 분류기(Naïve Bayes Classifier)

❖ 데이터 전처리

- 데이터를 토큰화한 후 BoW(Bag of Words)로 만들어 주어야 함

```
from sklearn.feature_extraction.text import CountVectorizer
dtmvector = CountVectorizer()
X_train_dtm = dtmvector.fit_transform(newsdata.data)
```

❖ Scikit-Learn에서 제공하는 나이브 베이즈 모델

- 나이브 베이즈 분류 수행

```
from sklearn.naive_bayes import MultinomialNB  # 다항분포 나이브 베이즈 모델
model = MultinomialNB()
model.fit(X_train_dtm, newsdata.target)
```

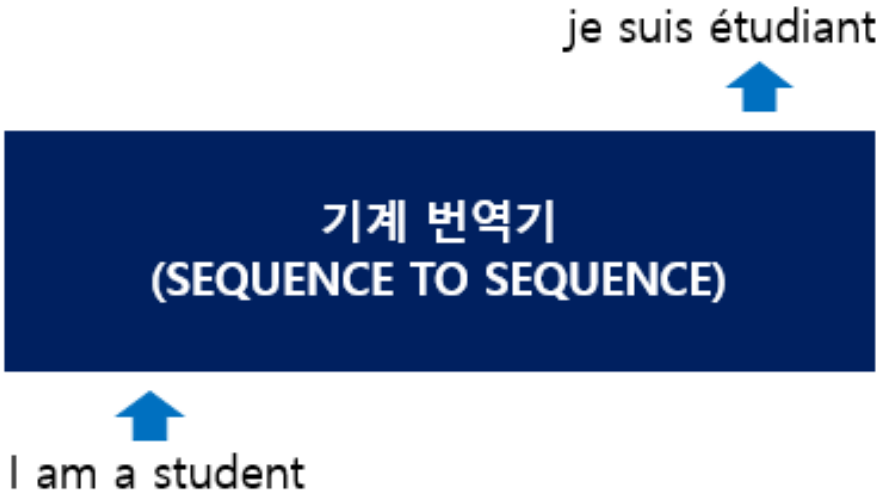
- 정확도 측정

```
from sklearn.metrics import accuracy_score     # 정확도 계산
predicted = model.predict(X_test_dtm) #테스트 데이터에 대한 예측
print("정확도: %.4f" % accuracy_score(newsdata_test.target, predicted))
# 예측값과 실제값 비교
```

■ Sequence-to-Sequence

❖ Sequence-to-Sequence 모델

- 입력된 시퀀스로부터 다른 도메인의 시퀀스를 출력하는 다양한 분야에서 사용되는 모델
- 입력 시퀀스 – 질문, 출력 시퀀스 – 대답 → 챗봇(Chatbot)
- 입력 시퀀스 – 입력 문장, 출력 시퀀스 – 번역 문장 → 번역기(Machine Translation)
- 내용 요약(Text Summarization), STT(Speech to Text) 등에도 사용



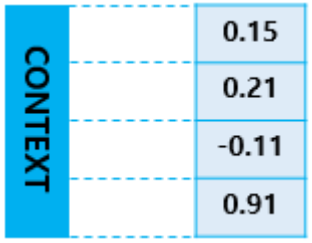
■ Sequence-to-Sequence

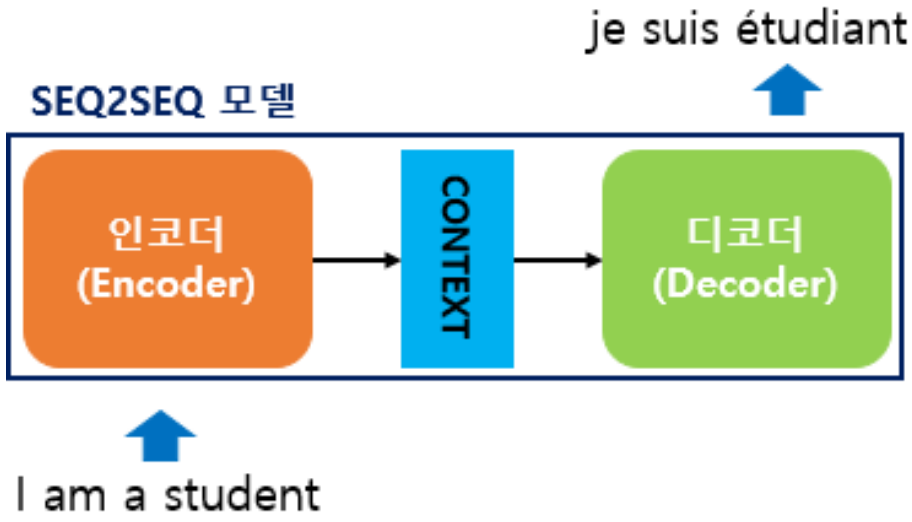
❖ 인코더(Encoder)

- 입력 문장의 모든 단어들을 순차적으로 입력받은 후
- 이 모든 단어 정보들을 압축해서 하나의 컨텍스트 벡터(Context vector)로 만듦
- 입력 문장이 컨텍스트 벡터로 압축되면 인코더는 컨텍스트 벡터를 디코더로 전송

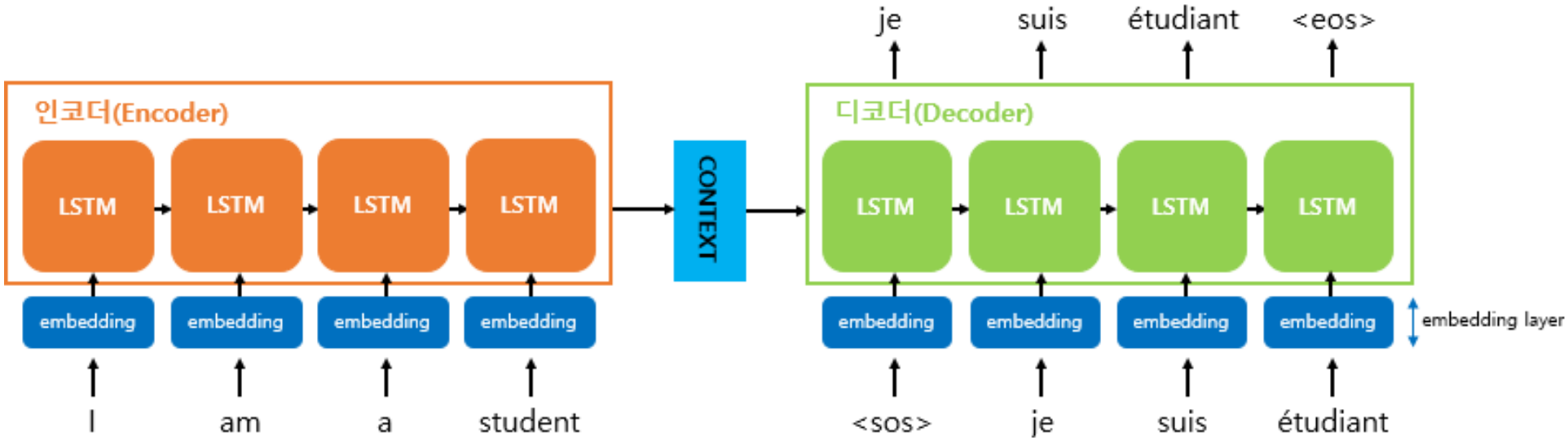
❖ 디코더(Decoder)

- 컨텍스트 벡터를 받아서 번역된 단어를 한 개씩 순차적으로 출력





■ Sequence-to-Sequence



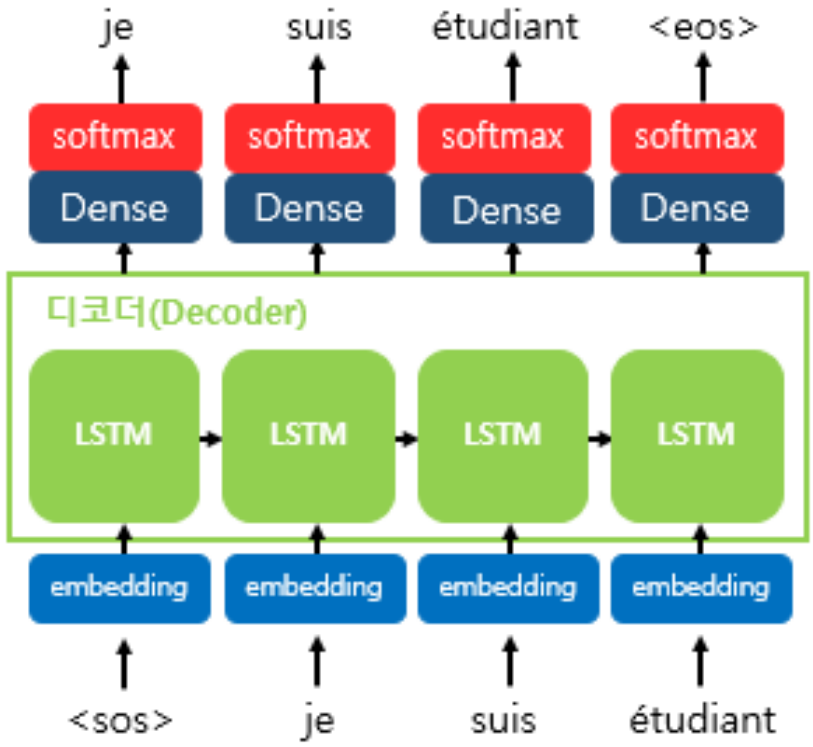
I	0.157	0.78	0.75	0.88
	-0.25	0.29	-0.81	-0.17
	0.478	-0.96	0.96	0.29
	-0.78	0.52	0.12	0.48
		am	a	student

<Embedding layer>

■ Sequence-to-Sequence

❖ 디코더(Decoder)

- 인코더의 마지막 RNN 셀의 은닉 상태인 컨텍스트 벡터를 첫번째 은닉 상태의 값으로 사용
- 첫번째 RNN 셀은 이 첫번째 은닉 상태의 값과, 현재 t에서의 입력값인 <sos>로부터, 다음에 등장할 단어를 예측
- 출력 단어로 나올 수 있는 단어들은 다양한 단어들이 있고 이를 예측하기 위해서 소프트맥스 함수 사용
- 각 시점(time step)의 RNN 셀에서 출력 벡터가 나오면, 해당 벡터는 소프트맥스 함수를 통해 출력 시퀀스의 각 단어별 확률값을 반환하고, 디코더는 출력 단어를 결정



■ 문자 레벨 기계 번역기(Character-Level Neural Machine Translation)

- 참조: sequence-to-sequence 10분만에 이해하기
  - 케라스 개발자 프랑수아 숄레의 블로그
  - 훈련 데이터로 병렬 코퍼스(parallel corpus)가 필요
- fra.txt
  - Watch me. Regardez-moi !
  - 왼쪽의 영어 문장과 오른쪽의 프랑스어 문장 사이에 탭으로 구분되는 구조가 하나의 샘플
  - 약 17만개의 병렬 문장 샘플을 포함

▶ 41_문자 레벨 기계 번역기.ipynb

■ Attention Mechanism

❖ RNN에 기반한 Sequence-to-Sequence 모델의 문제

- 하나의 고정된 크기의 벡터에 모든 정보를 압축하려고 하니까 정보 손실이 발생
- RNN의 고질적인 문제인 기울기 소실(Vanishing Gradient) 문제가 존재

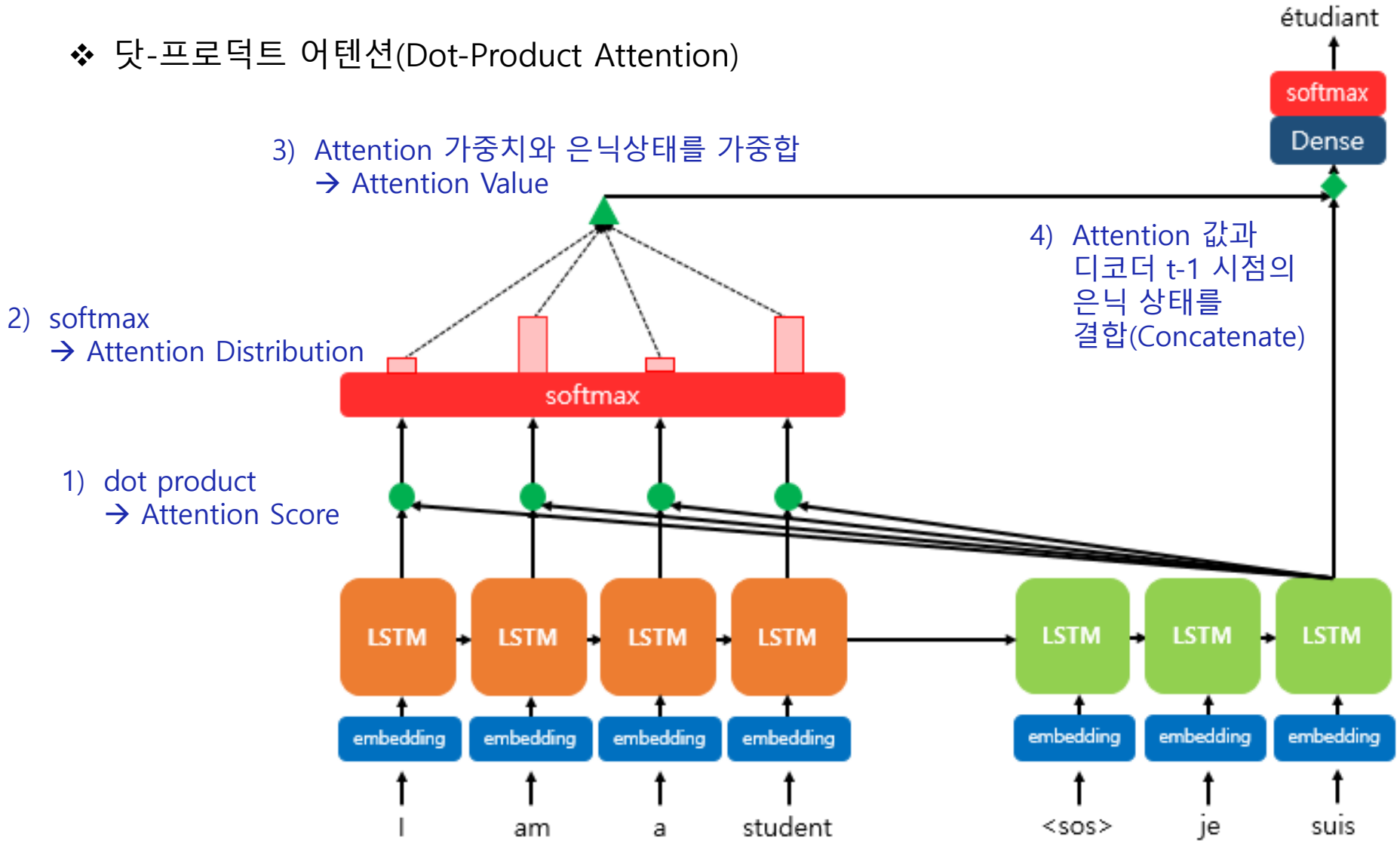
❖ Attention

- 입력 시퀀스가 길어지면 출력 시퀀스의 정확도가 떨어지는 것을 보정해주기 위한 기법
- 디코더에서 출력 단어를 예측하는 매 시점(time step)마다, 인코더에서의 전체 입력 문장을 다시 한 번 참고한다는 점
- 전체 입력 문장을 전부 다 동일한 비율로 참고하는 것이 아니라, 해당 시점에서 예측해야할 단어와 연관이 있는 입력 단어 부분을 좀 더 집중(attention)해서 보게 됨

$$\boldsymbol{Attention(Q, K, V) = Attention\ Value}$$

■ Attention Mechanism

❖ 닷-프로덕트 어텐션(Dot-Product Attention)



■ Attention Mechanism

❖ 닷-프로덕트 어텐션(Dot-Product Attention)

1) 어텐션 스코어(Attention Score)를 구한다.

$$score(s_{t-1}, h_i) = (s_{t-1})^T h_i$$

$$e^t = [(s_{t-1})^T h_i, \dots, (s_{t-1})^N h_N]$$

2) 소프트맥스(softmax) 함수를 통해 어텐션 분포(Attention Distribution)를 구한다.

$$\alpha^t = softmax(e^t)$$

3) 각 인코더의 어텐션 가중치와 은닉 상태를 가중합하여 어텐션 값(Attention Value)을 구한다. (Context Vector)

$$a_t = \sum_{i=1}^{N} (\alpha_i)^t \, h_i$$

4) 어텐션 값과 디코더의 t-1 시점의 은닉 상태를 결합한다.(Concatenate)

$$s_t = f(s_{t-1}, y_{t-1}, a_t)$$

$$s_t = f(v_t, y_{t-1})$$

■ Attention Mechanism

❖ 양방향 LSTM과 어텐션 메커니즘(BiLSTM with Attention mechanism) 사례

- IMDB 리뷰 감성 분류
- 바다나우 어텐션(Bahdanau Attention) 사용
  - 닷 프로덕트 어텐션의 스코어 함수

$$score(query, key) = query^T key$$

  - 바다나우 어텐션의 스코어 함수

$$score(query, key) = V^T \tanh(W_1 key + W_2 query)$$

- 텍스트 분류에서 어텐션 메커니즘을 사용하는 이유
  - RNN의 마지막 은닉 상태는 예측을 위해 사용되나
  - 마지막 은닉 상태는 몇 가지 유용한 정보들을 손실한 상태임.
  - RNN이 time step을 지나며 손실했던 정보들을 다시 참고하고자 함

▶ 42_BiLSTM with Attention mechanism-IMDB.ipynb