Category 4
Natural Language Processing

NLP 분류 모델용
데이터 세트 분석

sarcasm.json 데이터

- **sarcasm.json** 파일은 는 26,709 개의 기사의 '**headline**' 으로 구로 구성된 Text 데이터 세트이다

- '**is_sarcastic**'값이 **0**과 **1**의 label을 같는다
  0은 sarcastic(비판적) 이지 않은 텍스트
  1은 sarcastic(비판적) 인 텍스트
  binary classification model 용

sarcasm.json 데이터

[

{"article_link": "https://www.huffingtonpost.com/entry/versace-black-code_us_5861fbefe4b0de3a08f600d5", "headline": **"former versace store clerk sues over secret 'black code' for minority shoppers"**, "is_sarcastic": **0**},

{"article_link": "https://www.huffingtonpost.com/entry/roseanne-revival-review_us_5ab3a497e4b054d118e04365", "headline": **"the 'roseanne' revival catches up to our thorny political mood, for better and worse"**, "is_sarcastic": **0**},

{"article_link": "https://local.theonion.com/mom-starting-to-fear-son-s-web-series-closest-thing-she-1819576697", "headline": **"mom starting to fear son's web series closest thing she will have to grandchild"**, "is_sarcastic": **1**},

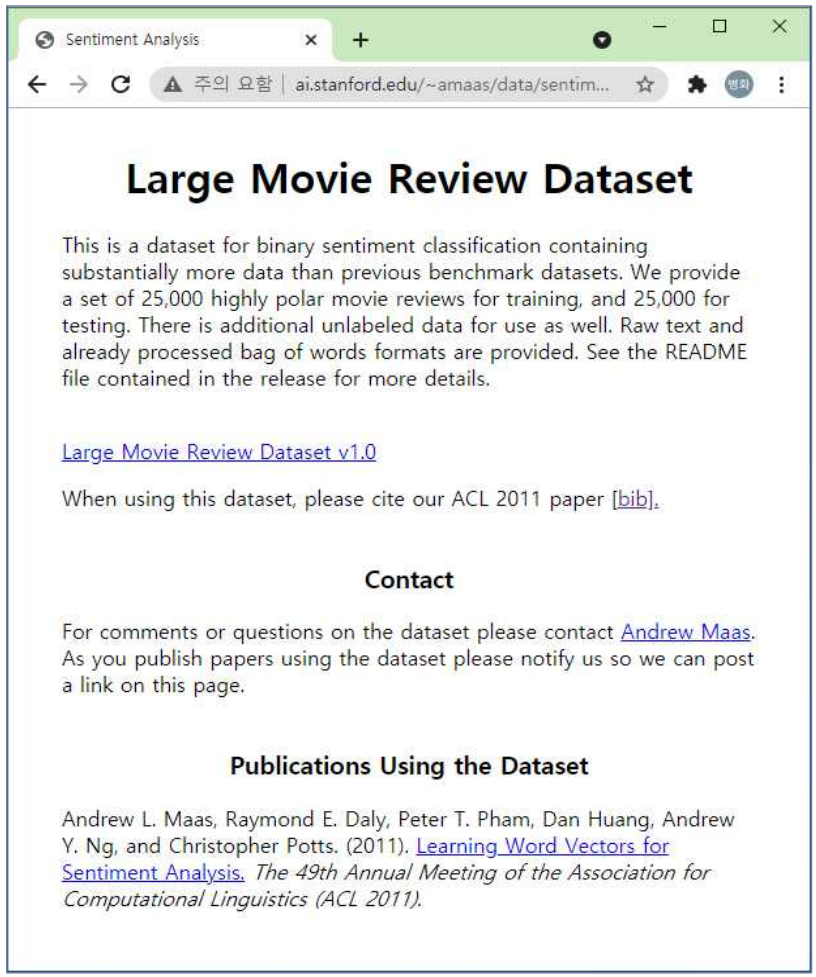
...

imdb_reviews

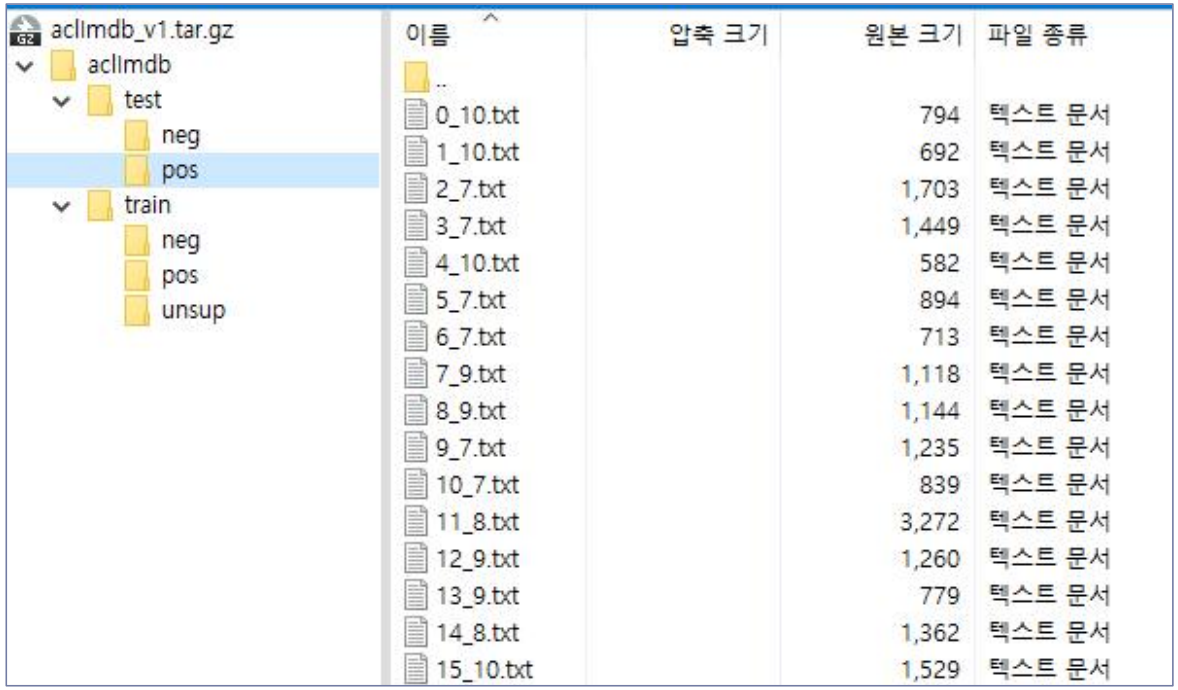
- **imdb_review** 데이터는 영화 감상평으로 train 25,000 개의 test 25,000 개로 구성된 Text 데이터 세트이다

- **'neg'** 와 **'pos'** 디렉토리에 각각 12,500개 씩 문장이 들어있다
  neg 는 **부정**인 텍스트
  pos 는 **긍정**인 텍스트
  binary classification model 용
  원본에는 label이 따로 없고 tfds으로 로딩 시 neg는 0
  pos는 1의 label 값이 추가 된다

imdb_reviews



다운로드된 imdb 압축파일 내용



imdb_reviews

- **train/pos/0_9.txt 파일 내용**

  Bromwell High is a cartoon comedy. It ran at the same time as some other programs about school life, such as "Teachers". My 35 years in the teaching profession lead me to believe that Bromwell High's satire is much closer to reality than is "Teachers". The scramble to survive financially, the insightful students who can see right through their pathetic teachers' pomp, the pettiness of the whole situation, all remind me of the schools I knew and their students. When I saw the episode in which a student repeatedly tried to burn down the school, I immediately recalled ..... at ..... High. A classic line: INSPECTOR: I'm here to sack one of your teachers. STUDENT: Welcome to Bromwell High. I expect that many adults of my age think that Bromwell High is far fetched. What a pity that it isn't!

irish-lyrics

- **irish-lyrics-eof.txt** 데이터는 train과 test 데이터가 따로 없는 **1692** 개의 라인 으로 구성된 순수한 Text 데이터이다. 문장 생성용

Come all ye maidens young and fair
And you that are blooming in your prime
Always beware and keep your garden fair
Let no man steal away your thyme
For thyme it is a precious thing
And thyme brings all things to my mind
nlyme with all its flavours, along with all its joys
Thyme, brings all things to my mind
Once I and a bunch of thyme
i thought it never would decay
Then came a lusty sailor

자연어처리(NLP) 개념

자연어 처리(NLP)란

- 한국어와 영어 등 우리가 평소에 쓰는 말을 자연어라고 한다.

- 자연어 처리(Natural Language Processing)를 풀어서 말하면
  '우리의 말을 컴퓨터에게 이해시키기 위한 기술(분야)'이다.

- 자연어 처리가 추구하는 목표는 사람의 말을 부드럽게
  컴퓨터가 이해하도록 만들어서,
  컴퓨터가 우리에게 도움이 되는 일을 수행하게 하는 것이다

단어의 의미

- 우리의 말은 '문자로'로 구성되며, 말의 의미는 '단어'로 구성된다.

- 단어는 의미의 최소 단위이기 때문에 자연어를 컴퓨터에게 이해시키는 데는 '단어의 의미'를 이해시키는 것이 중요하다.

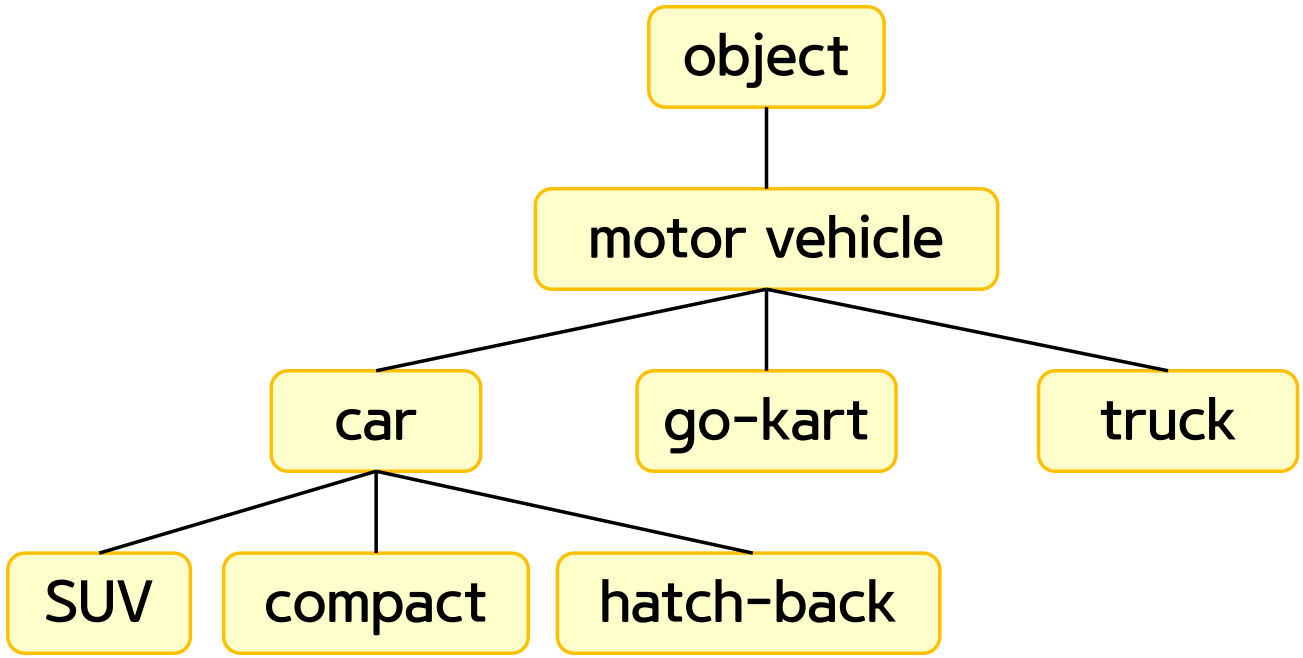
- 세가지 기법
  - 시소러스를 활용한 기법
  - 통계 기반 기법
  - 추론(예측) 기반 기법(word2vec)

시소러스

동의어의 예: "car", "auto", "automobile" 등은 "자동차"를 뜻하는 동의어다.

car = auto, automobile, machine, motorcar

단어들의 의미의 상/하위 관계에 기초해 그래프로 표현한다.



시소러스의 문제점

- WordNet과 같은 시소러스에는 수많은 단어에 대한 동의어와 계층 구조 등의 관계가 정의되어 있다. 그리고 이 지식을 이용하면 '단어의 의미'를 컴퓨터에 전달 할 수 있다. 하지만 사람이 수작업으로 레이블링하는 방식에는 문제들이 존재한다.

- **시대 변화에 대응하기 어렵다.**
  신조어 혹은 의미 변화된 단어들을 바로 적용 시키기 어렵다.

- **사람을 쓰는 비용이 든다.**
  현존하는 영어 단어의 수는 1,000만 개가 넘으며 WordNet에 등록된 단어는 20만 개 이상이다.

- **단어의 미묘한 차이를 표현할 수 없다.**
  가령, 빈티지와 레트로의 의미는 같으나 용법의 차이가 존재한다.
  위 문제점들을 피하기 위해 '통계 기반 기법'과 신경망을 사용한 '추론 기반 기법‘을 사용한다

단어의 분산 표현

- 색에는 고유한 이름이 붙여진 다채로운 색들도 있고,
  RGB(Red/Green/Blue)라는 세가지 성분이 어떤 비율로 섞여 있느냐로 표현
  하는 방법이 있다.
  전자는 색의 가짓수만큼 의 이름을 부여하는 반면에 후자는 색을 3차원의
  벡터로 표현한다.

- 여기서 주목할 점은 RGB같은 벡터 표현이 단 3개의 성분으로 간결하게 표현할 수 있고, 색을 더 정확하게 명시할 수 있다는 점이다.

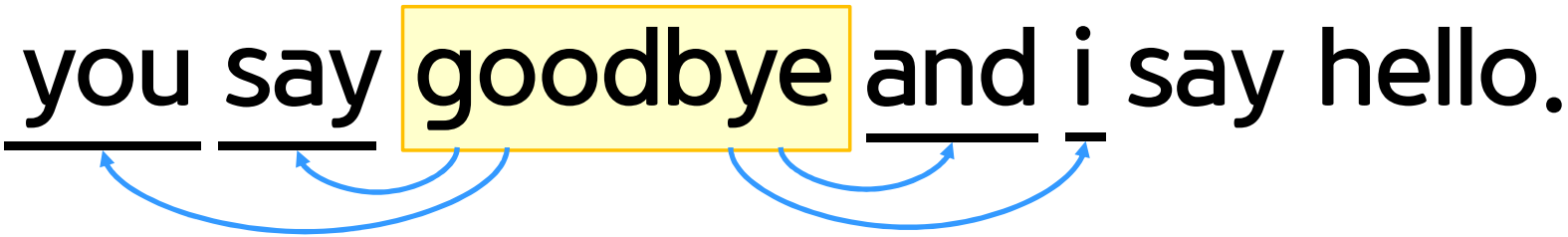
- '색'을 벡터로 표현하듯 '단어'도 벡터로 표현할 수 있다. 이를 단어의 '분산 표현'이라고 한다.

분포 가설

분포 가설이란 단어의 의미는 주변 단어에 의해 형성된다는 것이다.
분포 가설이 말하고자 하는 것은 단어 자체에는 의미가 없고,
그 단어가 사용된 '맥락'이 의미를 형성한다는 것이다.

예를 들어, I drink beer를 I guzzle beer라고 해도 guzzle을 drink로 이해할 수 있다는 것이다.

윈도우 크기가 2인 '맥락'의 예. 단어 "goodbye"에 주목한다면,
그 좌우의 두 단어(총 네 단어)를 맥락으로 이용한다.



위 그림에서 goodbye를 기준으로 좌우의 두 단어씩이 '맥락'에 해당한다.
맥락의 크기를 '윈도우 크기'라고 한다. 여기서는 '위도우 크기'가 2이기 때문에
좌우로 두 단어씩이 맥락에 포함된다.

동시 발생 행렬

분포 가설에 기초해 단어를 벡터로 나타내는 방법을 생각해보면
주변 단어를 세어보는 방법이 떠오를 것이며 이를 '통계 기반'기법이라고 한다.

단어 "you"의 맥락을 세어본다.

단어가 총 7개이며 윈도우 크기는 1로 하고 단어 ID가 0인 'you'부터
단어의 맥락에 해당하는 단어의 빈도를 세어보겠다.
'you'의 맥락은 'say'라는 단어 하나뿐이다.

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0

동시 발생 행렬

모든 단어 각각의 맥락에 해당하는 단어의 빈도를 세어 표로 정리 한다.

	you	say	goodbye	and	i	hello	.
you	0	1	0	0	0	0	0
say	1	0	1	0	1	1	0
goodbye	0	1	0	1	0	0	0
and	0	0	1	0	1	0	0
i	0	1	0	1	0	0	0
hello	0	1	0	0	0	0	1
.	0	0	0	0	0	1	0

위의 표는 모든 단어에 대해 동시발생하는 단어를 표에 정리한 것이다.
위 표의 각 행은 벡터이며 행렬의 형태를 띄어 동시발생 행렬이라 한다.

통계 기반 기법의 문제점

통계 기반 기법과 추론 기반 기법 비교

통계 기반 기법은 학습 데이터를 한꺼번에 처리한다.

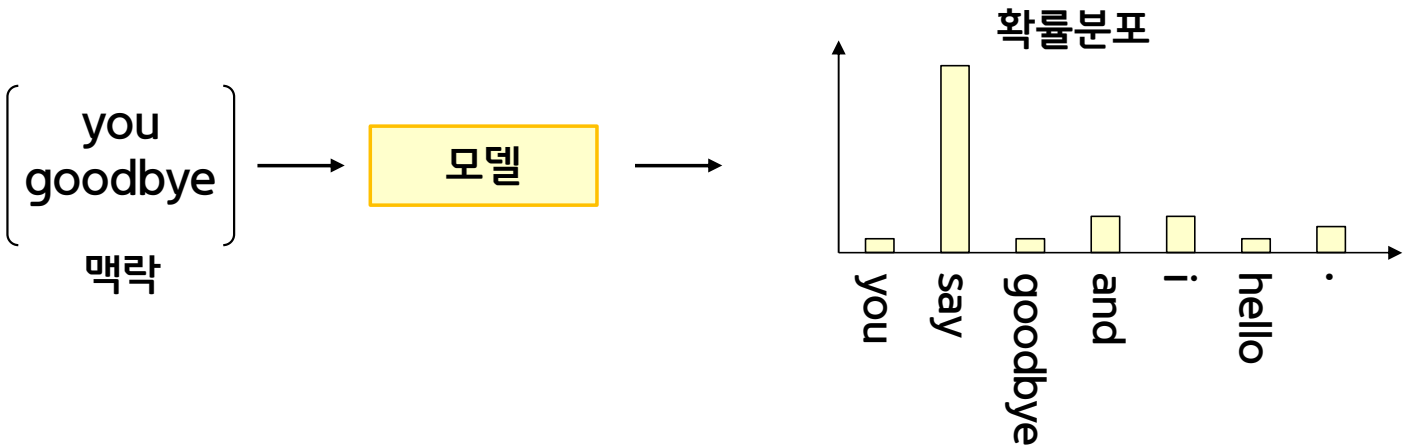
배치학습 추론 기반 기법은 학습 데이터의 일부를 사용하여 순차적으로 학습한다.

미니배치 학습 :
말뭉치의 어휘 수가 많아 SVD 등 계산량이 큰 작업을 처리하기
어려운 경우에도 신경망을 학습시킬 수 있다는 의미이다.
데이터를 작게 나눠 학습하기 때문이다.

추론 기반 기법 개요

모델 관점에서 보면, 추론 문제는 다음과 같다.

추론 기반 기법: 맥락을 입력하면 모델은 각 단어의 출현 확률을 출력한다.



추론 기반 기법에는 어떠한 모델이 등장한다.
우리는 이 모델로 신경망을 사용한다.
모델은 맥락 정보를 입력 받아 출현할 수 있는 각 단어의 출현 확률을 출력한다.
이러한 틀 안에서 말뭉치를 사용해 모델이 올바른 추측을 내놓도록 학습시킨다.
그리고 그 학습의 결과로 단어의 분산 표현을 얻는 것이 추론 기반 기법의 전체 그림이다.

신경망에서의 단어 처리

지금부터 신경망을 이용해 단어를 처리해보자.
단어를 있는 그대로 처리할 수 없으니 고정 길이의 벡터로 변환해야 한다.
이때 사용하는 대표적인 방법이 단어를 원 핫 표현으로 변환하는 것이다.
원 핫 표현이란, 벡터의 원소 중 하나만 1이고, 나머지는 모두 0인 벡터를 말한다.

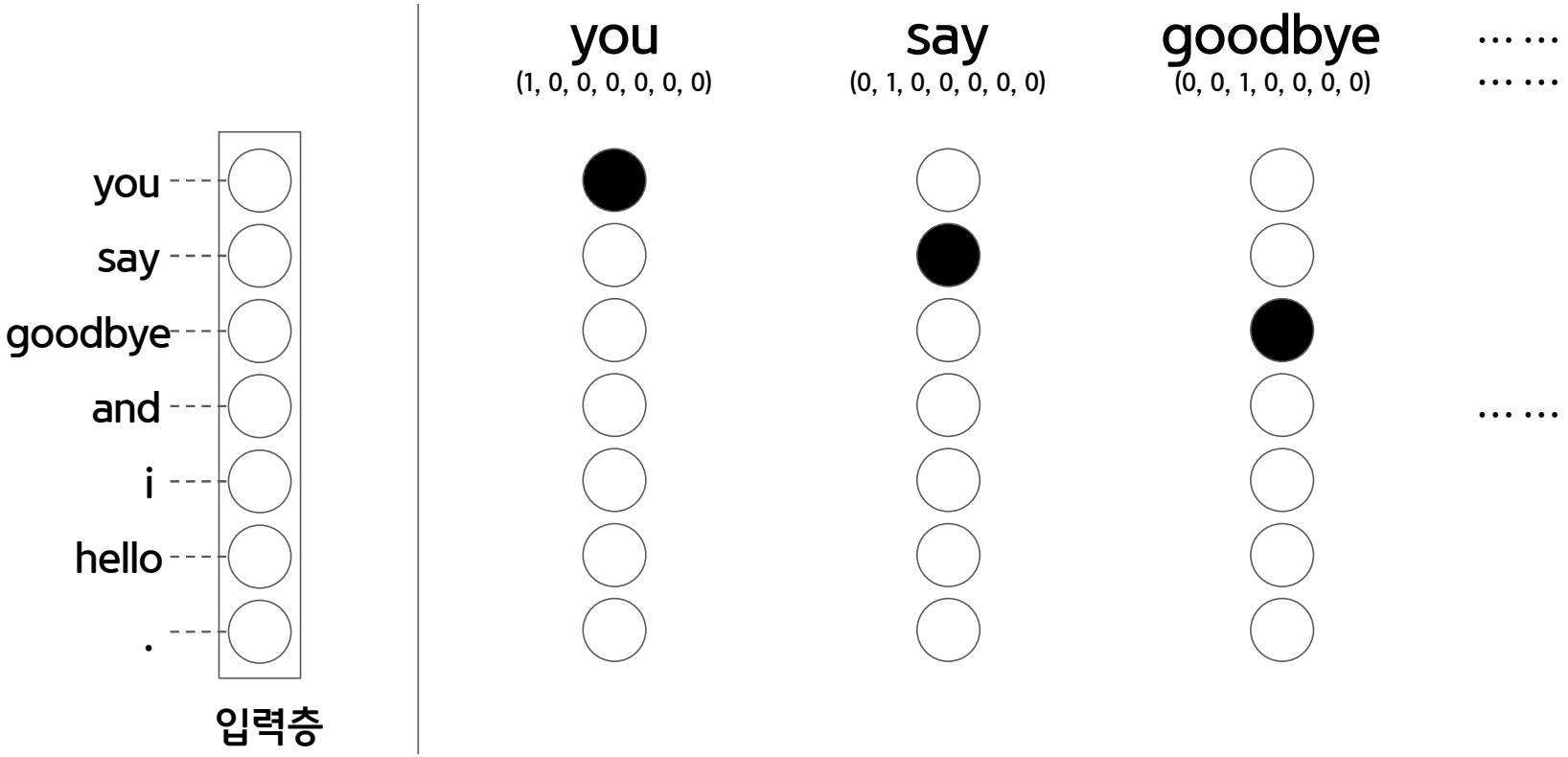
단어(텍스트)	단어 ID	원 핫 표현
you goodbye	0 2	(1, 0, 0, 0, 0, 0, 0) (0, 0, 1, 0, 0, 0, 0)

단어를 원 핫 표현으로 변환하는 방법

- 먼저 총 어휘 수만큼의 원소를 갖는 벡터를 준비하고,
- 인덱스가 단어 ID 와 같은 원소를 1로, 나머지는 모두 0으로 설정한다.
  이처럼 단어를 고적 길이 벡터로 변환하면, 신경망의 입력층은 뉴런의 수를 고정할 수 있다.

신경망에서의 단어 처리

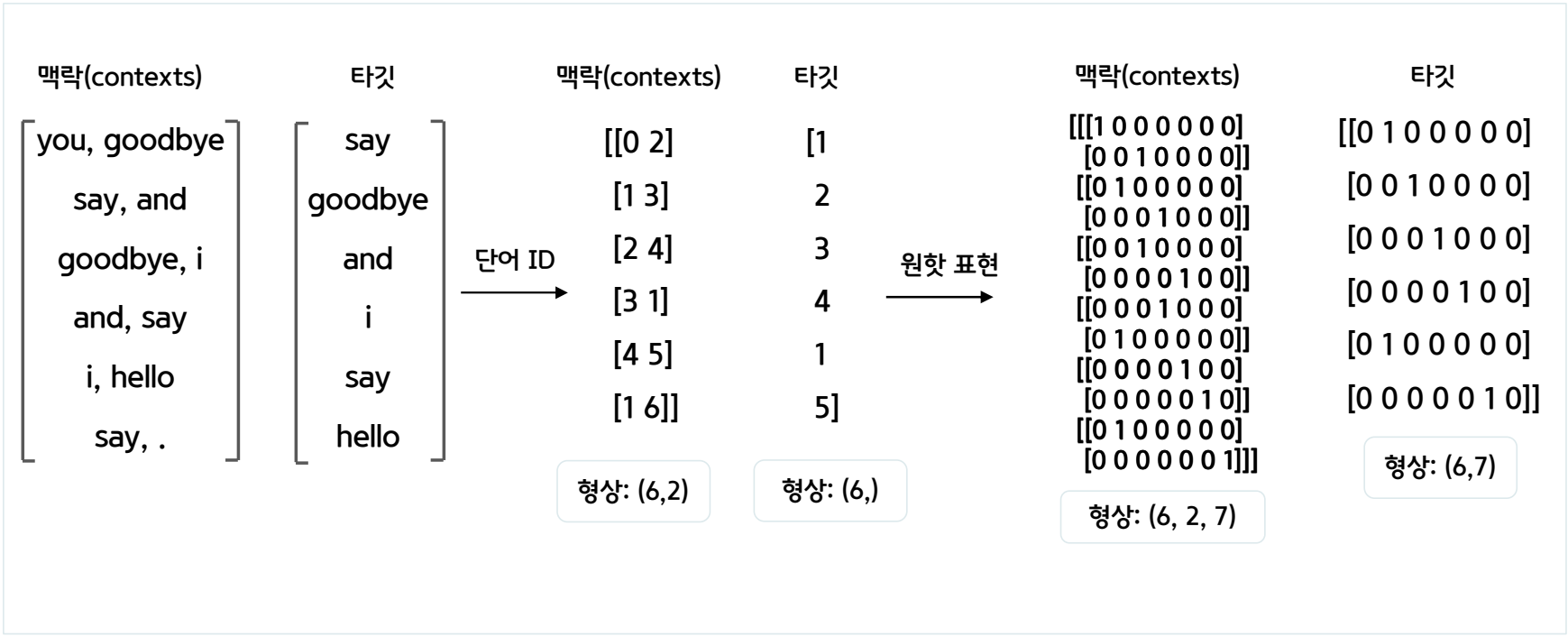
입력층의 뉴런: 각 뉴런이 각 단어에 대응한다.(해당 뉴런이 1이면 검은색, 0이면 흰색)



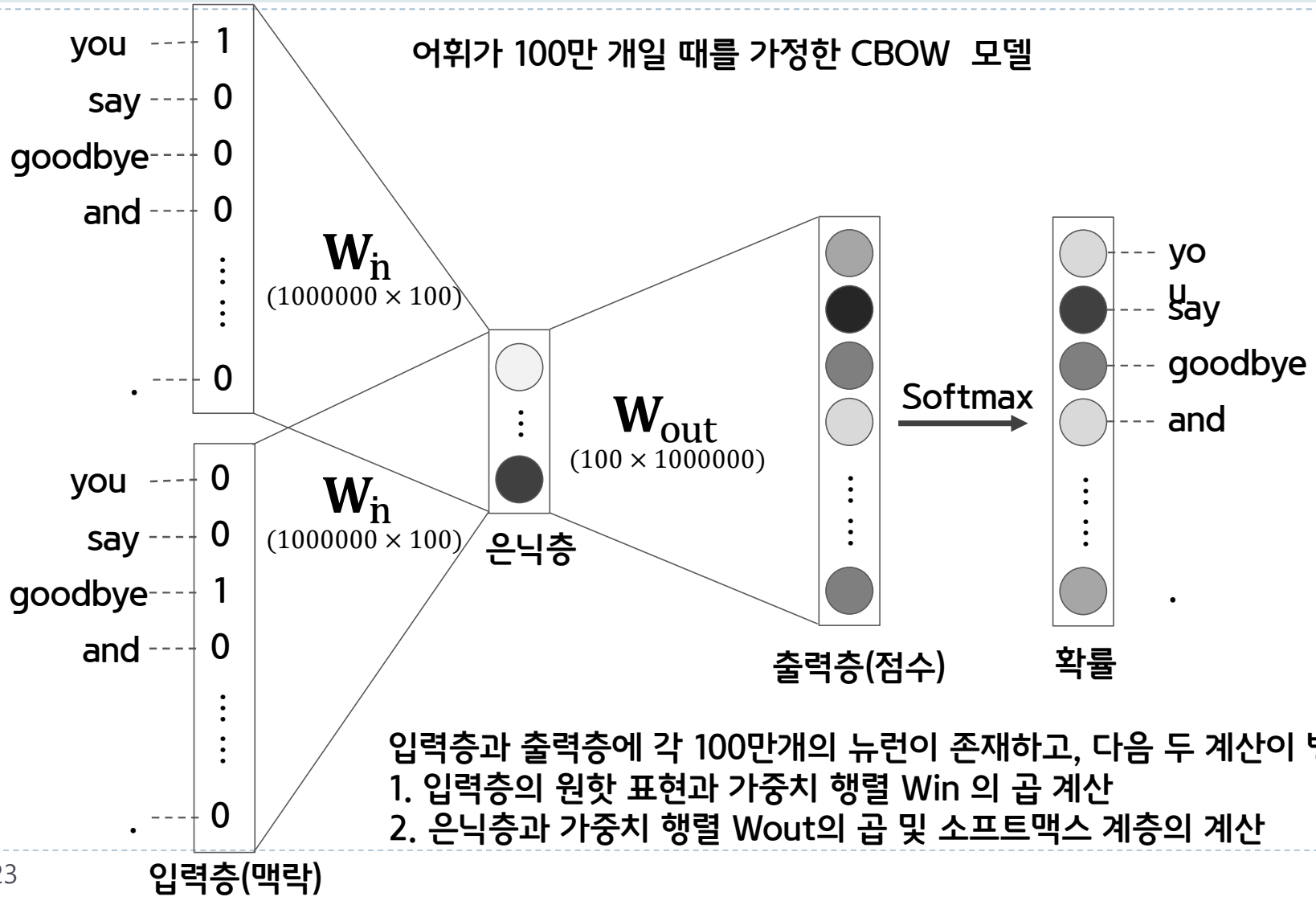
단어를 벡터로 나타낼 수 있고, 신경망을 구성하는 계층들은 벡터를 처리할 수 있다.
다시 말해, 단어를 신경망으로 처리할 수 있다는 뜻이다.

맥락과 타깃

맥락과 타깃을 원핫 표현으로 변환하는 예



word2vec : CBOW



입력층과 출력층에 각 100만개의 뉴런이 존재하고, 다음 두 계산이 병목된다.
1. 입력층의 원핫 표현과 가중치 행렬 Win 의 곱 계산
2. 은닉층과 가중치 행렬 Wout의 곱 및 소프트맥스 계층의 계산

Embedding 계층

Embedding 이란, 텍스트를 구성하는 하나의 단어를 수치화하는 방법의 일종이다.
텍스트 분석에서 흔히 사용하는 방식은 단어 하나에 인덱스 정수를 할당하는 Bag of Words 방법이다.
이 방법을 사용하면 문서는 단어장에 있는 단어의 갯수와 같은 크기의 벡터가 되고 단어장의
각 단어가 그 문서에 나온 횟수만큼 벡터의 인덱스 위치의 숫자를 증가시킨다.
단어장이 "I", "am", "a", "boy", "girl" 다섯개의 단어로 이루어진 경우 각 단어에 다음과 같이 숫자를 할당한다.
"I": 0
"am": 1
"a": 2
"boy": 3
"girl": 4

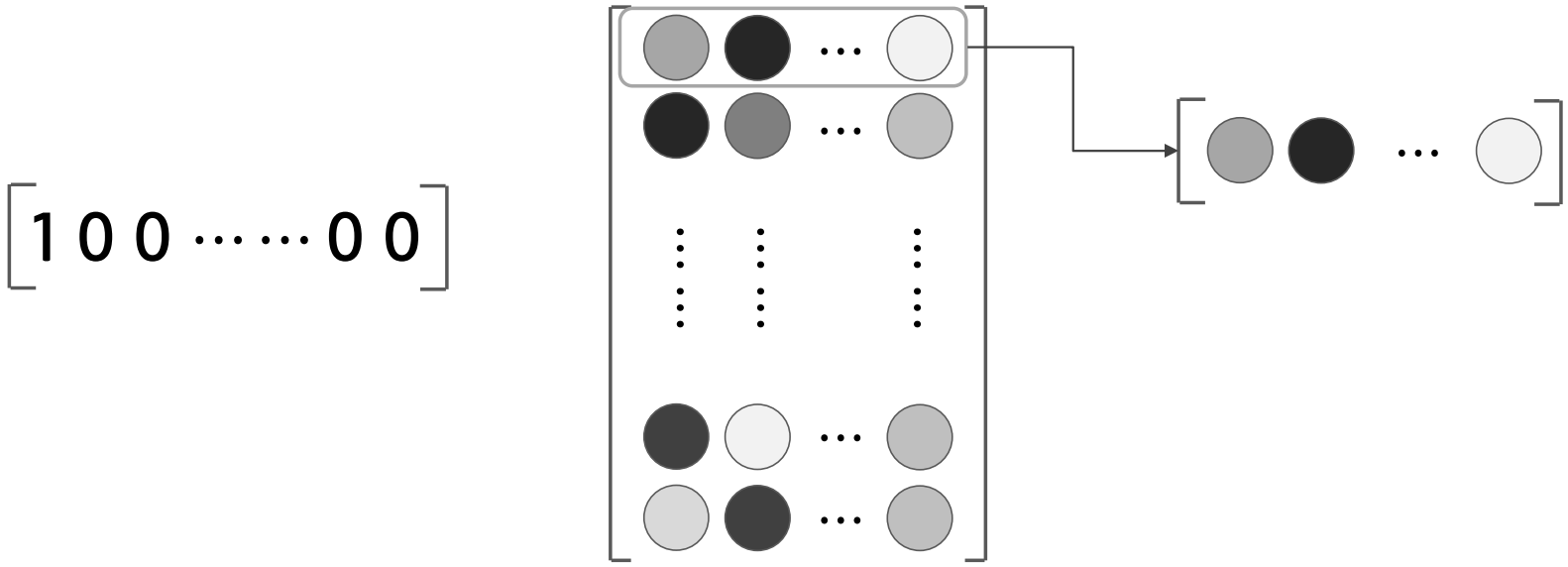
이 때 "I am a girl" 이라는 문서는 다음과 같이 벡터로 만들 수 있다.

[1, 1, 1, 0, 1]

단어 임베딩은 하나의 단어를 하나의 인덱스 정수가 아니라 실수 벡터로 나타낸다.
예를 들어 2차원 임베딩을 하는 경우 다음과 같은 숫자 벡터가 될 수 있다.
"I": (0.3, 0.2)
"am": (0.1, 0.8)
"a": (0.5, 0.6)
"boy": (0.2, 0.9)
"girl": (0.4, 0.7)

Embedding 계층

맥락(원핫 표현)과 MatMul 계층의 가중치를 곱한다.



$$\underset{(1\times1000000)}{c} \qquad \underset{(1000000\times100)}{W_{in}} = \underset{(1\times100)}{h}$$

Word Embedding

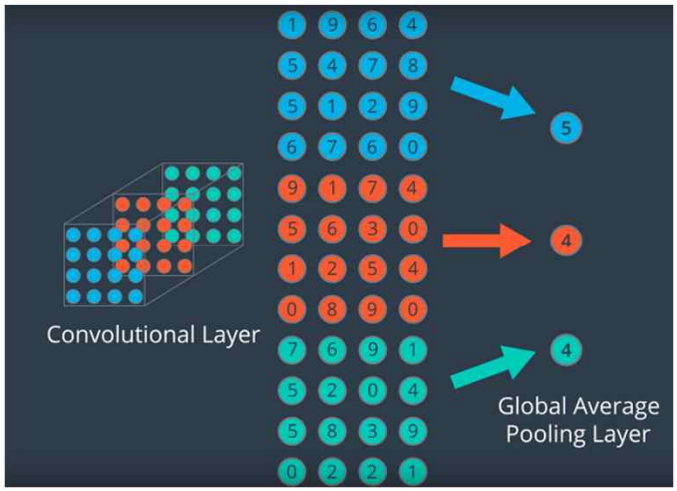
- 단어를 밀집 벡터(dense vector)의 형태로 표현하는 방법을 **워드 임베딩(word embedding)**이라고 한다. 이 밀집 벡터를 워드 임베딩 과정을 통해 나온 결과라고 하여 **임베딩 벡터(embedding vector)**라고도 합니다.

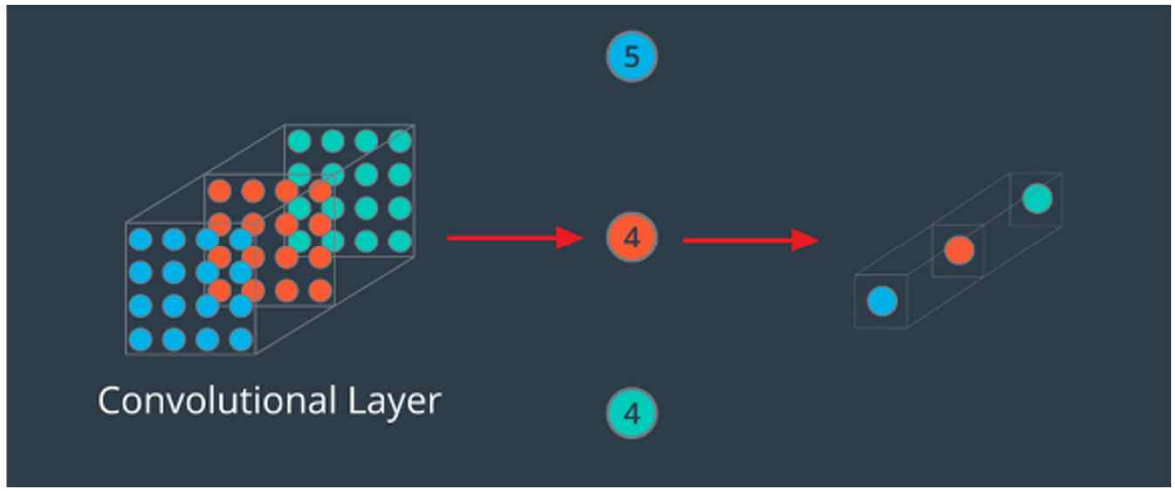
-	원-핫 벡터	임베딩 벡터
차원	고차원(단어 집합의 크기)	저차원
다른 표현	희소 벡터의 일종	밀집 벡터의 일종
표현 방법	수동	훈련 데이터로부터 학습함
값의 타입	1과 0	실수

https://wikidocs.net/22660

Global Average Pooling

• GAP(global average pooling)은 앞에서 설명한 Max(Average) Pooling 보다 더 급격하게 feature의 수를 줄입니다.
• GAP의 목적은 앞에서 사용한 Pooling과 다릅니다. GAP의 목적은 feature를 1차원 벡터로 만들기 위함 입니다.





https://gaussian37.github.io/dl-concept-global_average_pooling/

- **토큰화(Tokenization)**

  Tokenization이란 Text를 여러 개의 Token으로 나누는 것을 말합니다.
  보통 공백, 구두점, 특수문자 등으로 이를 나누는데 그 방법에 따라 다양한 Tokenizer 가 있다.

- 케라스 Tokenizer

```
tf.keras.preprocessing.text.Tokenizer(
    num_words=None,
    filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
    lower=True, split=' ', char_level=False, oov_token=None,
    document_count=0, **kwargs
)
```


https://zamezzz.tistory.com/314

텍스트 전처리 : 토큰화

```
# 가장 빈도가 높은 10000개의 단어들만 사용하여 토큰화
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)

# 단어 인덱스를 구축
tokenizer.fit_on_texts(training_sentences)

word_index = tokenizer.word_index
# print(word_index)

# 문자열을 정수 인덱스의 리스트로 변환 : 정수 인코딩
training_sequences = tokenizer.texts_to_sequences(training_sentences)  # type은 list

# 패딩, 벡터 표현을 얻음 : 신경망에 입력할 x값
training_padded = pad_sequences(training_sequences, maxlen=max_length, padding=padding_type, truncating=

# test 데이터 : 정수 인덱스의 리스트로 변환
testing_sequences = tokenizer.texts_to_sequences(testing_sentences)

# test 데이터 : 벡터 표현을 얻음
testing_padded = pad_sequences(testing_sequences, maxlen=max_length, padding=padding_type, truncating=t
```

텍스트 전처리

- 자연어 처리의 전처리 단계 : **토큰화, 단어 집합 생성, 정수 인코딩, 패딩**
- **Tokenizer 정의**

  tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)

  num_words : 단어 집합 max사이즈를 지정. 가장 빈도수가 높은 단어만 사용

  oov_token : 단어 집합에 없는 단어를 어떻게 표기할 것인지 지정.

- **패딩(Padding)**

  training_padded = pad_sequences(training_sequences, maxlen=max_length, padding=padding_type, truncating=trunc_type)

  maxlen : 최대 문장 길이를 정의. 최대 문장길이보다 길면, 잘라낸다

  truncating : 문장의 길이가 maxlen보다 길 때 앞을 자를지 뒤를 자를지 정의

  padding : 문장의 길이가 maxlen보다 짧을 때 채워줄 값을 앞을 채울지, 뒤를 채울지 정의

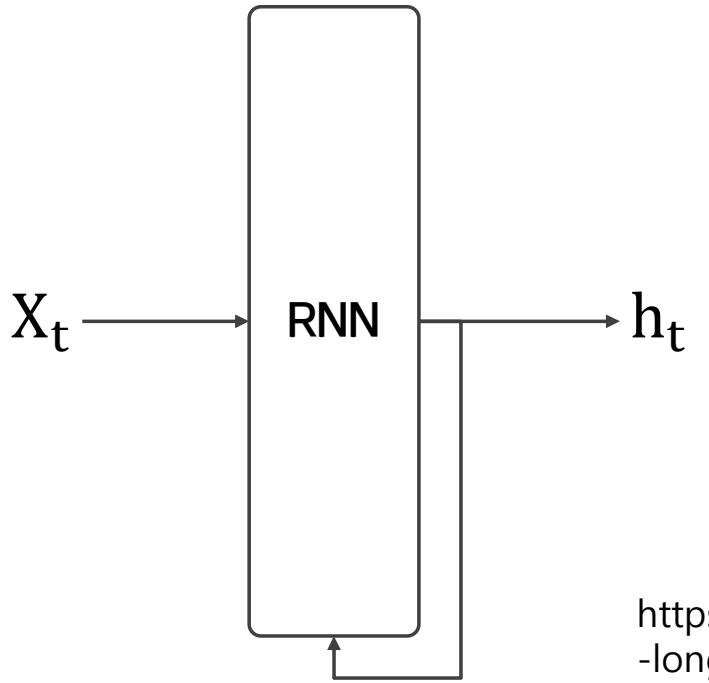
https://katie0809.github.io/2020/06/22/tf-study4/

순환신경망(RNN)

순환하는 신경망(RNN , Recurrent Neural Network)

순환하기 위해서는 닫힌 경로가 필요하다.
닫힌 경로 혹은 순환하는 경로가 존재해야 데이터가 같은 장소를 반복해 왕래할 수 있고
데이터가 순환하면서 과거의 정보를 기억하는 동시에 최신 데이터로 갱신 될 수 있다.

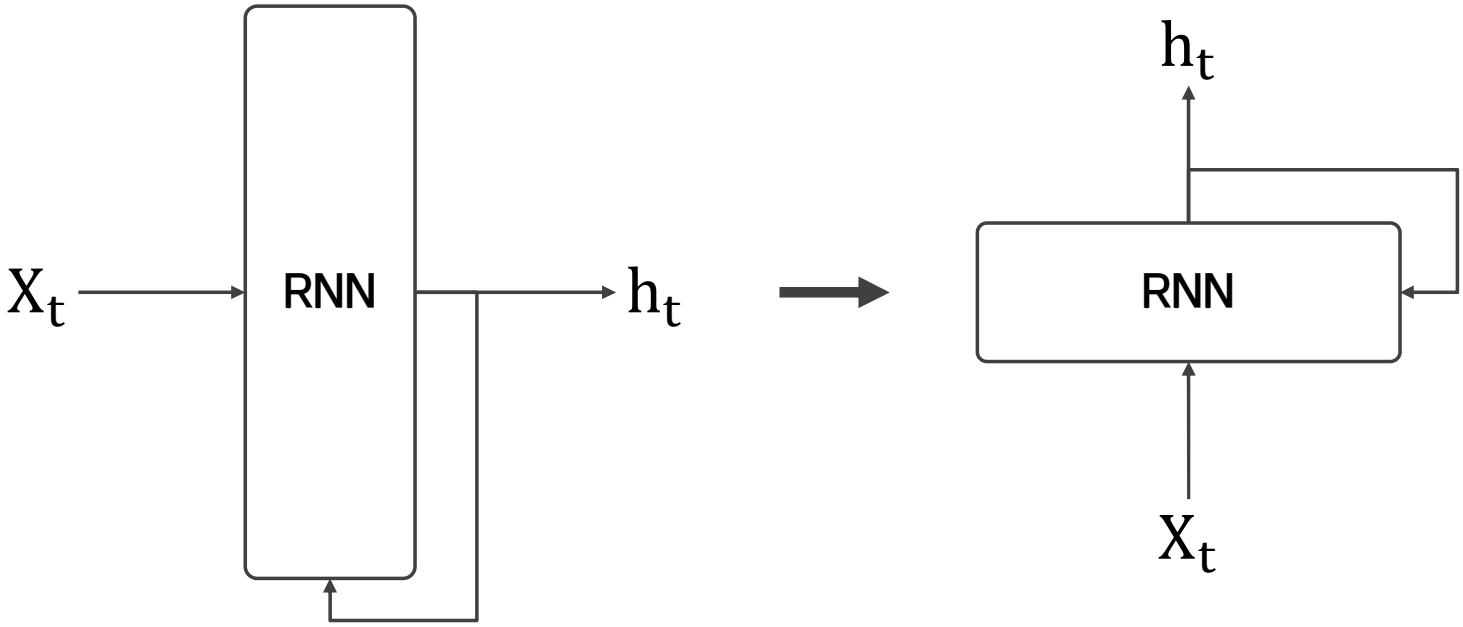
순환 경로를 포함하는 RNN 계층



https://dgkim5360.tistory.com/entry/understanding-long-short-term-memory-lstm-kr

순환하는 신경망(RNN , Recurrent Neural Network)

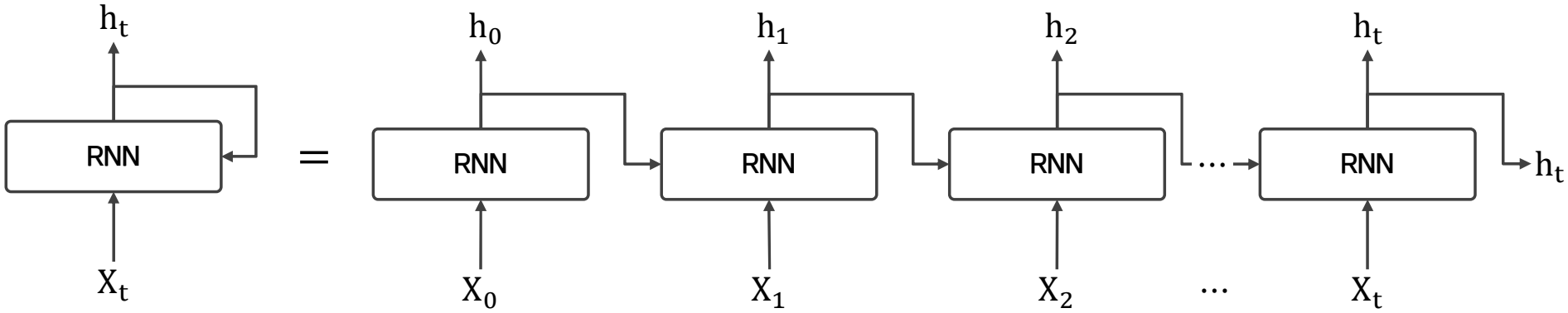
계층을 90도 회전시켜 그린다.



t: 시각
시계열 데이터(x0, x1,...xt, ...)가 RNN계층에 입력되고 이에 대응해 (h0, h1, ..., ht, ...)가 출력된다.
각 시각에 입력되는 xt를 벡터라고 가정했을 때
문장(단어 순서)을 다루는 경우를 예로 든다면 각 단어의 분산 표현(단어 벡터)이 xt가 되며 이 분산 표현이 순서대로 하나씩 RNN계층에 입력된다.

순환하는 신경망(RNN , Recurrent Neural Network)

RNN 계층의 순환 구조 펼치기



RNN계층의 순환 구조를 펼침으로써 오른쪽으로 성장하는 긴 신경망으로 변신
피드포워드 신경망(데이터가 한 방향으로만 흐른다)과 같은 구조이지만 위 그림에서는
다수의 RNN계층 모두가 실제로는 '같은 계층'인 것이 지금까지의 신경망과는 다른 점이다.

각 시각의 RNN계층은 그 계층으로의 입력과 1개 전의 RNN계층으로부터의 출력을 받는데
이 두 정보를 바탕으로 현 시각의 출력을 계산한다.

순환하는 신경망(RNN , Recurrent Neural Network)

순환 구조 펼치기

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

Wx: 입력 x를 출력 h로 변환하기 위한 가중치
Wh: 1개의 RNN출력을 다음 시각의 출력으로 변환하기 위한 가중치
b: 편향
h(t-1), xt: 행벡터

ht는 다른 계층을 향해 위쪽으로 출력되는 동시에 다음 시각의 RNN계층(자기 자신)을 향해 오른쪽으로도 출력된다.

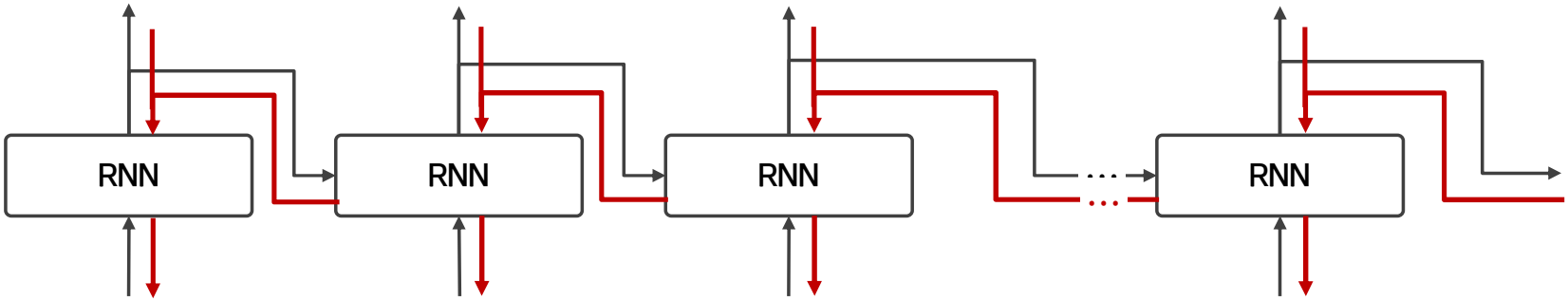
RNN의 출력 ht는 은닉상태(hidden state) 혹은 은닉 상태 벡터(hidden state vector)라고 한다.

RNN은 h라는 '상태'를 가지고 있으며 위의 식의 형태로 갱신된다고 해석할 수 있다.
RNN계층을 '상태를 가지는 계층' 혹은 '메모리(기억력)가 있는 계층'이라고 한다.

순환하는 신경망(RNN , Recurrent Neural Network)

BPTT

순환 구조를 펼친 RNN 계층에서의 오차역전파



순환 구조를 펼친 후의 RNN에는 (일반적인) 오차역전파법을 적용할 수 있다.
먼저 순전파를 수행하고 이어서 역전파를 수행하여 원하는 기울기를 구할 수 있다.
여기서의 오차역전파법은 '시간 방향으로 펼친 신경망의 오차역전파법'이란 뜻으로
BPTT(Backpropagation Through Time)라고 한다.

문제점

- 시계열 데이터의 시간 크기가 커지는 것에 비례하여 BPTT가 소비하는 컴퓨팅 자원도 증가
- 시간 크기가 커지면 역전파 시의 기울기가 불안정해짐

순환하는 신경망(RNN)

RNN 순환 구조 펼치기

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

Wx: 입력 x를 출력 h로 변환하기 위한 가중치
Wh: 1개의 RNN출력을 다음 시각의 출력으로 변환하기 위한 가중치
b: 편향
h(t-1), xt: 행벡터

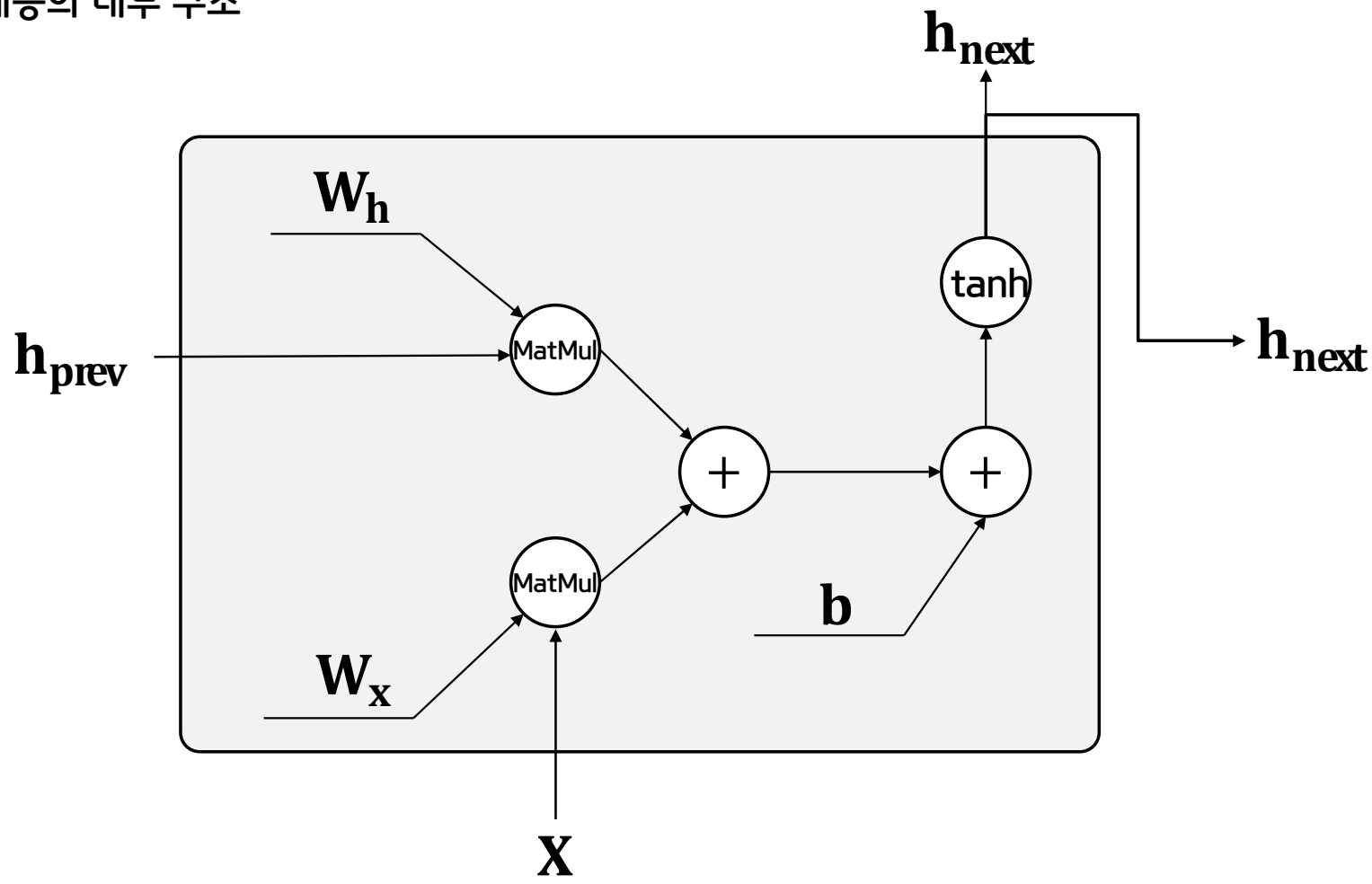
ht는 다른 계층을 향해 위쪽으로 출력되는 동시에 다음 시각의 RNN계층(자기 자신)을 향해 오른쪽으로도 출력된다.

RNN의 출력 ht는 은닉상태(hidden state) 혹은 은닉 상태 벡터(hidden state vector)라고 한다.

RNN은 h라는 '상태'를 가지고 있으며 위의 식의 형태로 갱신된다고 해석할 수 있다.
RNN계층을 '상태를 가지는 계층' 혹은 '메모리(기억력)가 있는 계층'이라고 한다.

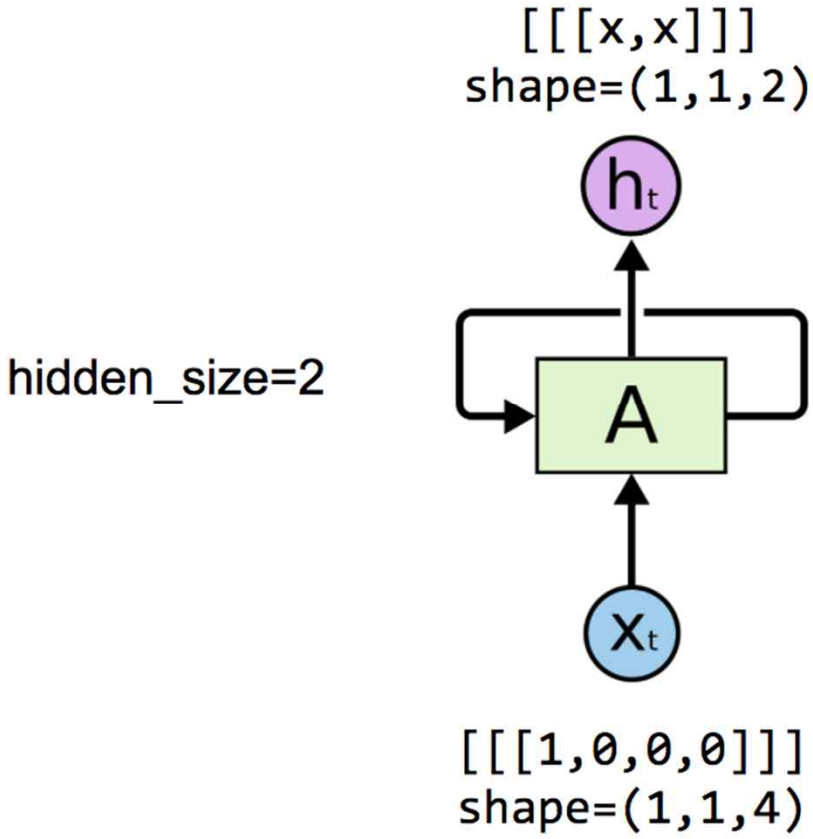
순환하는 신경망(RNN)

RNN 계층의 내부 구조



순환하는 신경망(RNN , Recurrent Neural Network)

RNN 입출력 구조




```
# One hot encoding
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

순환하는 신경망(RNN , Recurrent Neural Network)

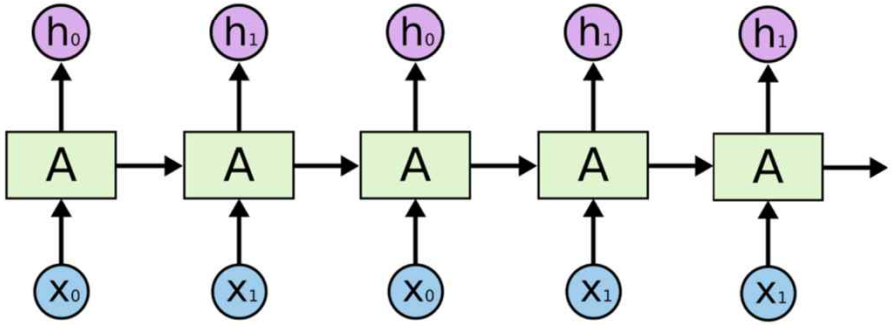
RNN 입출력 구조

hidden_size=2
sequacne_length=5
batch = 3

Using word vector

```
# One hot encoding
h = [1, 0, 0, 0]
e = [0, 1, 0, 0]
l = [0, 0, 1, 0]
o = [0, 0, 0, 1]
```

```
shape=(3,5,2): [[[x,x], [x,x], [x,x], [x,x], [x,x]],
                [[x,x], [x,x], [x,x], [x,x], [x,x]],
                [[x,x], [x,x], [x,x], [x,x], [x,x]]]
```



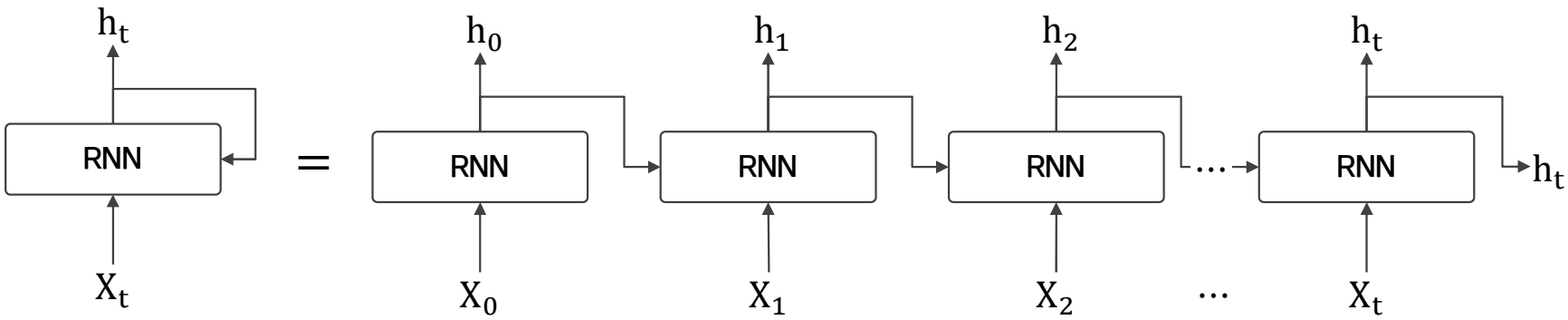

```
shape=(3,5,4): [[[1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,1,0], [0,0,0,1]], # hello
                [[0,1,0,0], [0,0,0,1], [0,0,1,0], [0,0,1,0], [0,0,1,0]]  # eolll
                [[0,0,1,0], [0,0,1,0], [0,1,0,0], [0,1,0,0], [0,0,1,0]]] # lleel
```

RNN의 문제점

RNN의 문제점

RNN은 시계열 데이터의 장기 의존 관계를 학습하기 어렵다.
그 이유는 BPTT에서 기울기 소실 혹은 기울기 폭발이 일어나기 때문이다.

RNN 계층 : 순환을 펼치기 전과 후



RNN의 문제점

기울기 소실 또는 **기울기 폭발**

"?"에 들어갈 단어는 ? : (어느 정도의)장기 기억이 필요한 문제의 예

Tom was watching TV in his room. Mary came into the room. Mary said hi to ?

언어 모델은 주어진 단어들을 기초로 다음에 출현할 단어를 예측하는 일을 한다.

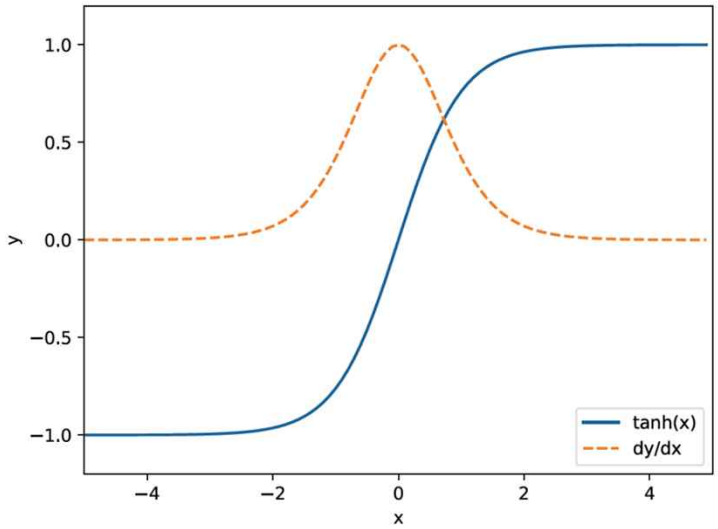
Tom was watching TV in his room. Mary came into the room. Mary said hi to ?
여기서 ?에 들어갈 단어는 Tom이다. RNNLM이 이 문제에 올바르게 답하라면,
현재 맥락에서 'Tom이 방에서 TV를 보고 있음'과 '그 방에 Mary가 들어옴'이란 정보를 기억해야 한다.

즉, 이런 정보를 RNN 계층의 은닉 상태에 인코딩해 보관해야 한다.

RNN의 문제점

기울기 소실 의 원인

y=tanh(x)의 그래프(점선은 미분)



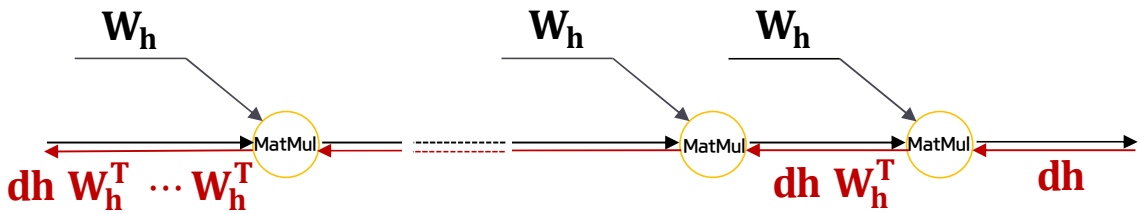
그림에서 점선이 y=tanh(x)의 미분이고 값은 1.0 이하이며, x가 0으로부터 멀어질수록 작아진다.
즉, 역전파에서 기울기가 tanh 노드를 지날 때마다 값은 계속 작아진다는 의미이다.
그리고 tanh 함수를 T번 통과하면 기울기도 T번 반복해서 작아진다.

'MatMul(행렬 곱)' 노드의 경우 tanh 노드를 무시하기로 한다.
그러면 RNN 계층의 역전파 시 기울기는 'MatMul' 연산에 의해서만 변화하게 된다.

RNN의 문제점

기울기 소실 또는 기울기 폭발의 원인

RNN 계층의 행렬 곱에만 주목했을 때의 역전파의 기울기



상류로부터 dh라는 기울기가 흘러온다고 가정하고 이때 MatMul 노드에서의 역전파는
dh(W_h^T)라는 행렬 곱으로 기울기를 계산한다.

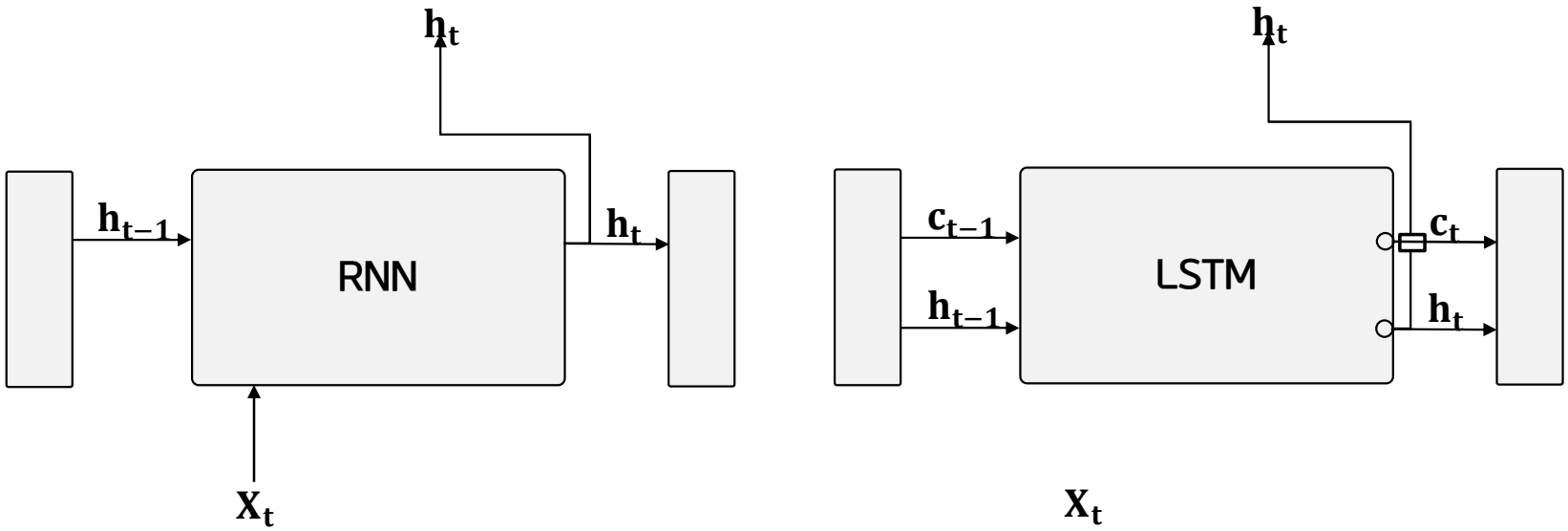
그리고 같은 계산을 시계열 데이터의 시간 크기만큼 반복한다.
주의할 점은 행렬 곱셈에서 매번 똑같은 W_h 가중치를 쓴다는 것이다.

즉, 행렬 곱의 기울기는 시간에 비례해 지수적으로 증가/감소함을 알 수 있으며
증가할 경우 기울기 폭발이라고 한다. 기울기 폭발이 일어나면 오버플로를 일으켜 NaN 같은 값을 발생시킨다.
반대로 기울기가 감소하면 기울기 소실이 일어나고 이는 일정 수준 이하로 작아지면 가중치 매개변수가
더 이상 갱신되지 않으므로 장기 의존 관계를 학습할 수 없게 된다.

LSTM
(Long Short Term Memory)

(LSTM , Long Short Term Memory)

RNN 계층과 LSTM 계층 비교



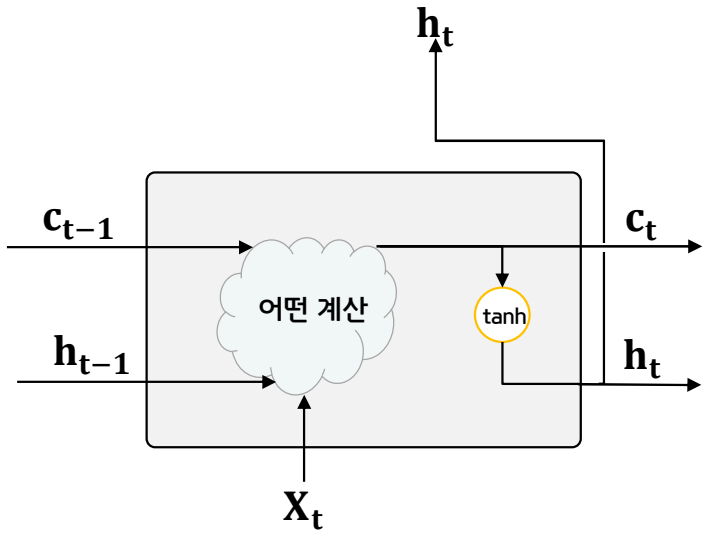
이 그림에서는 LSTM 계층의 인터페이스에는 c라는 경로가 있다는 차이가 있다.
여기서 c를 기억 셀이라 하며 LSTM 전용의 기억 메커니즘이다. **(LSTM은 게이트가 추가된 RNN)**

기억 셀의 특징은 데이터를 LSTM 계층 내에서만 주고받는다는 것이다.
다른 계층으로는 출력하지 않는다는 것이다. 반면, LSTM의 은닉 상태 h는
RNN 계층과 마찬가지로 다른 계층, 위쪽으로 출력된다.

(LSTM , Long Short Term Memory)

LSTM 계층 조립하기

기억 셀 c_t 를 바탕으로 은닉 상태 h_t 를 계산하는 LSTM 계층



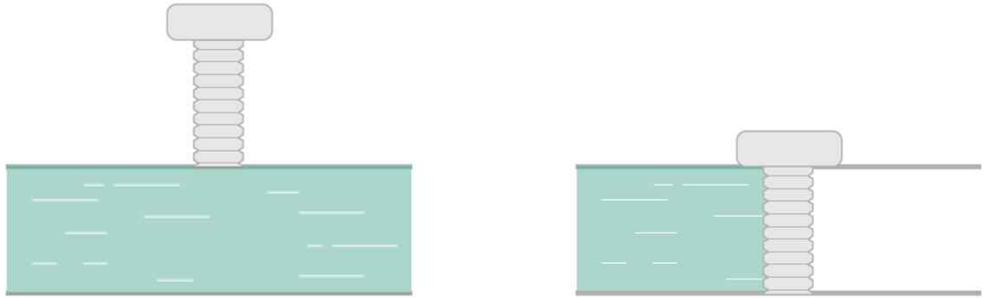
그림에서 현재의 기억 셀 c_t 는 3개의 입력 (c_{t-1}, h_{t-1}, X_t)으로부터
'어떤 계산'을 수행하여 구할 수 있다.
여기서 핵심은 갱신된 c_t 를 사용해 은닉 상태 h_t를 계산한다는 것이다.
또한 이 계산은 h_t =tanh(c_t)인데, 이는 c_t 의 각 요소에 tanh 함수를 적용한다는 뜻이다.

'게이트'란 우리나라로 '문'을 의미하는 단어이다.
문은 열거나 닫을 수 있듯이, 게이트는 데이터의 흐름을 제어한다.

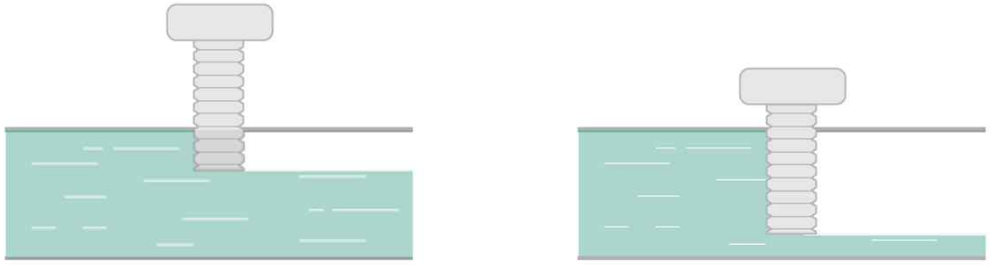
(LSTM , Long Short Term Memory)

LSTM 계층 조립하기

비유하자면 게이트는 물의 흐름을 제어한다.



물이 흐르는 양을 0.0~1.0 범위에서 제어한다.



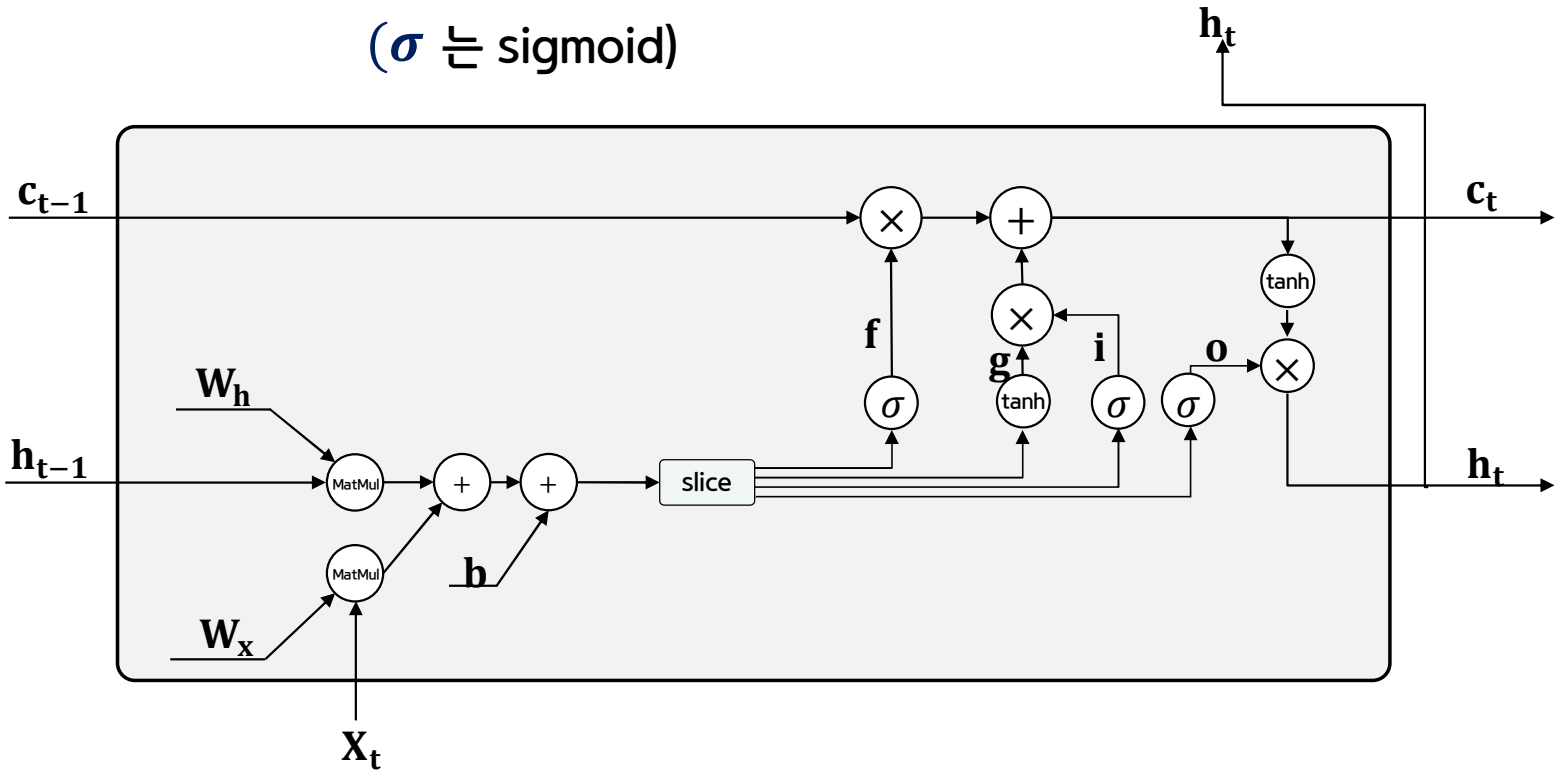
0.7(70%)　　0.2(20%)

(LSTM , Long Short Term Memory)

LSTM 구현

4개 분의 가중치를 모아 Matmul을 수행하는 LSTM의 계산 그래프

(σ 는 sigmoid)



만약 W_x, W_h, b 각각에 4개 분의 가중치가 포함되어 있다고 가정하면 위의 그림처럼 그래프가 그려진다.

(LSTM , Long Short Term Memory)

LSTM 구현

4개의 게이트 추가 f : forget g : get(공식 용어 아님) **i : input o : output**

$$f = \sigma(x_t W_x^{(f)} + h_{t-1} W_h^{(f)} + b^{(f)})$$

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

$$i = \sigma(x_t W_x^{(i)} + h_{t-1} W_h^{(i)} + b^{(i)})$$

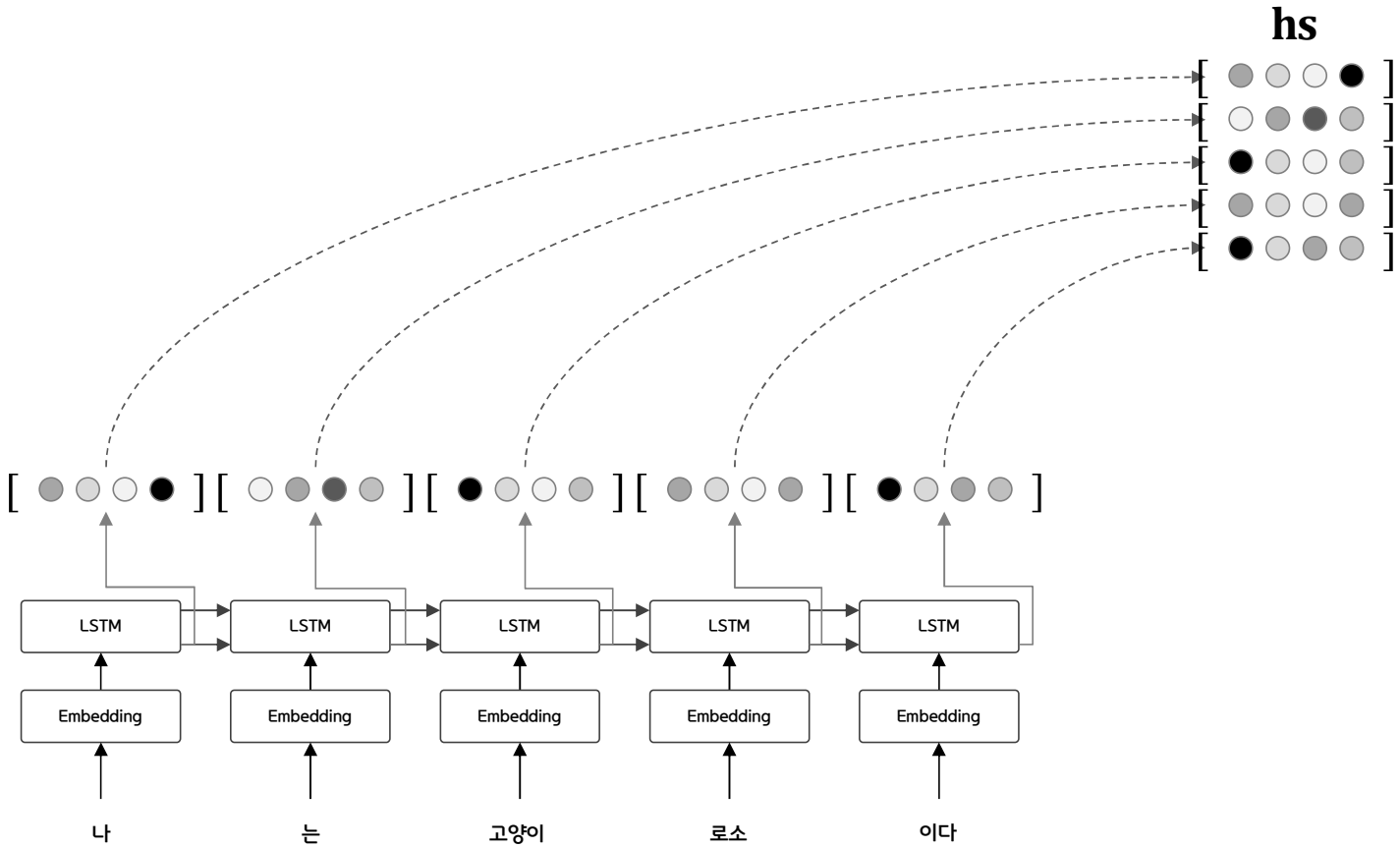
$$o = \sigma(x_t W_x^{(o)} + h_{t-1} W_h^{(o)} + b^{(o)})$$

$$c_t = f \odot c_{t-1} + g \odot i$$

$$h_t = o \odot \tanh(c_t)$$

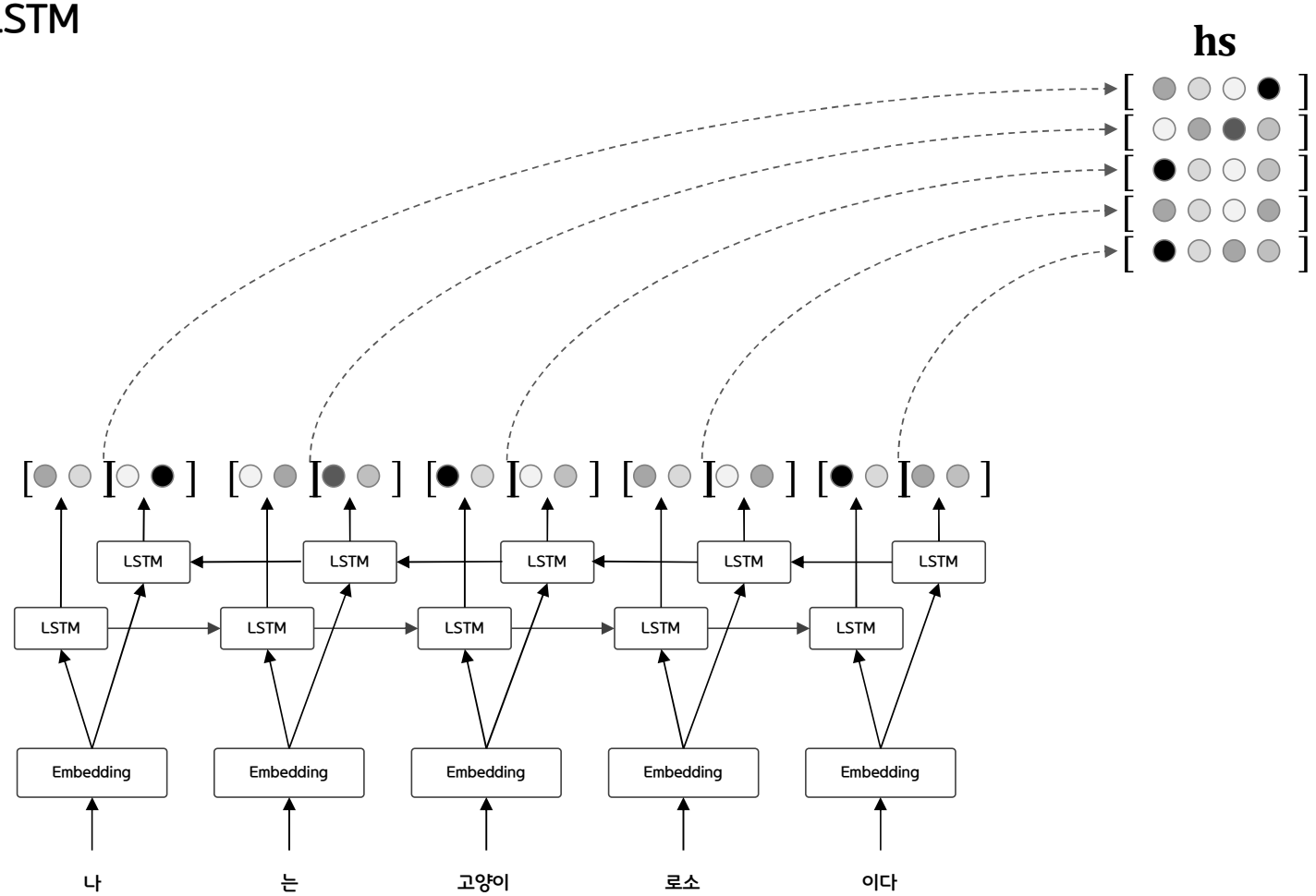
(LSTM , Long Short Term Memory)

단방향 LSTM



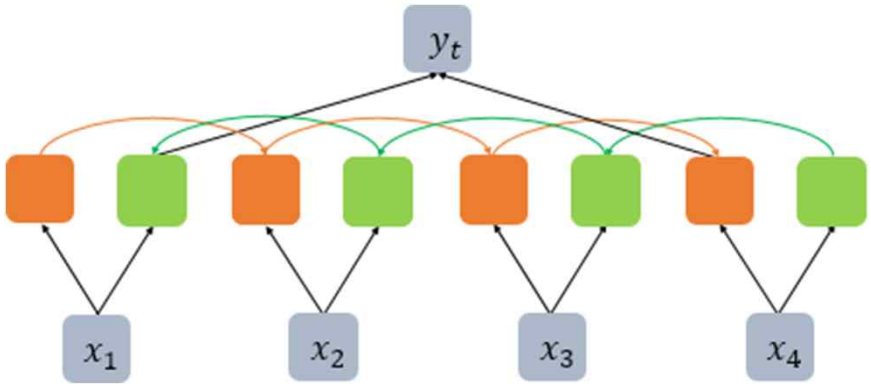
(LSTM , Long Short Term Memory)

양방향 LSTM

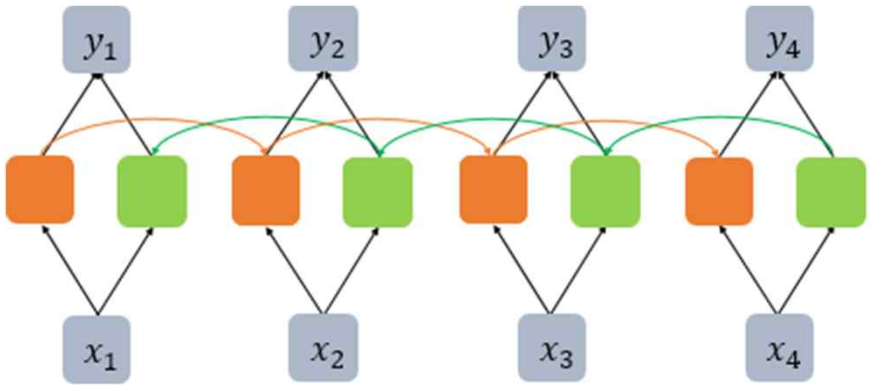


양방향 LSTM 출력

return_sequences=False인 경우



return_sequences=True인 경우



https://wikidocs.net/106473

양방향 LSTM 출력

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64, return_sequences=True)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

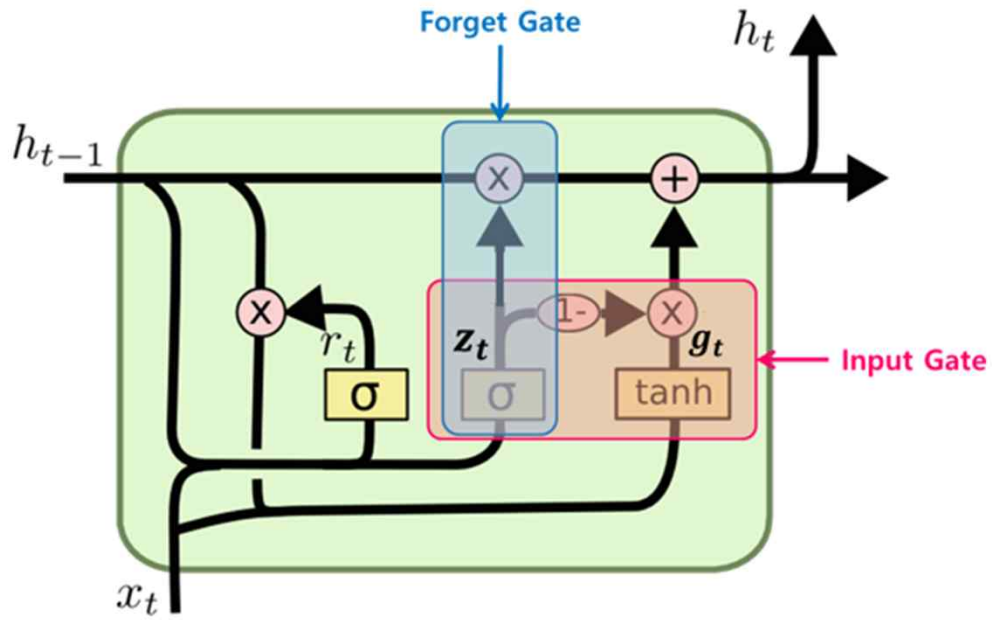
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 16)           160000
_________________________________________________________________
dropout (Dropout)            (None, 100, 16)           0
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 128)          41472
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 128)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)                41216
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 64)                4160
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 65
=================================================================
Total params: 246,913
Trainable params: 246,913
Non-trainable params: 0
_________________________________________________________________
```

양방향 LSTM은 출력이 단방향 일때의 2배의 크기가 된다

GRU(Gated Recurrent Unit)

GRU(Gated Recurrent Unit) 셀은 2014년에 K. Cho(조경현) 등에 의해 논문에서 제안된 LSTM 셀의 **간소화된 버전**이라고 할 수 있으며, 다음의 그림과 같은 구조를 가진다

논문 url : https://arxiv.org/pdf/1406.1078v3.pdf



https://excelsior-cjh.tistory.com/185

Text 생성
(Text Generation)

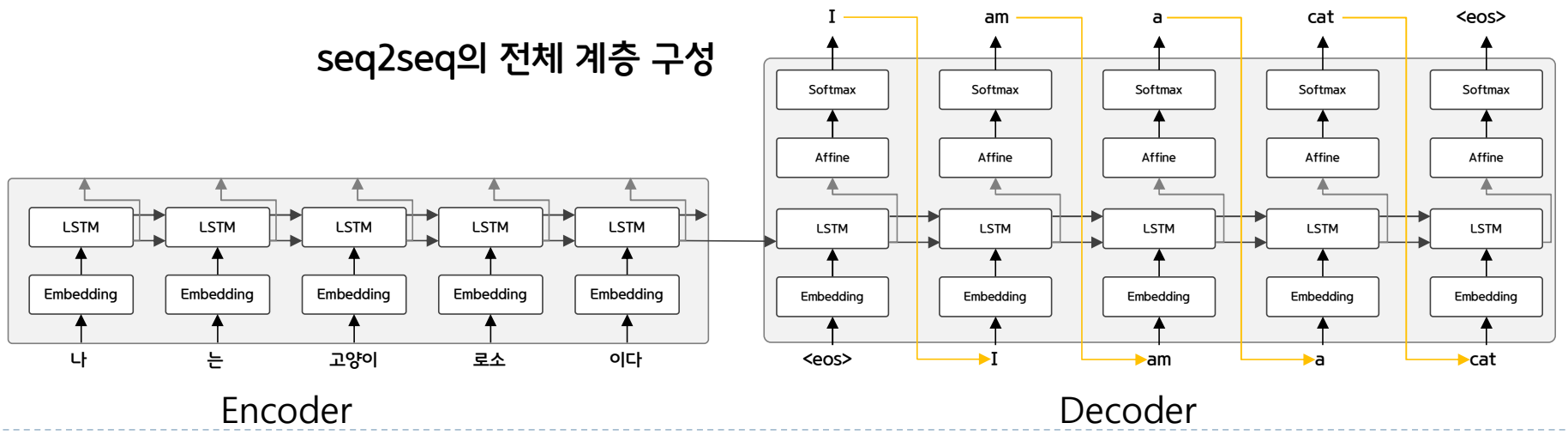
seq2seq 의 원리

seq2seq 는 LSTM 두 개로 구성된다.
(Encoder 의 LSTM, Decoder 의 LSTM)

이때 LSTM 계층의 은닉 상태가 Encoder 와 Decoder 를 이어주는 가교가 된다.

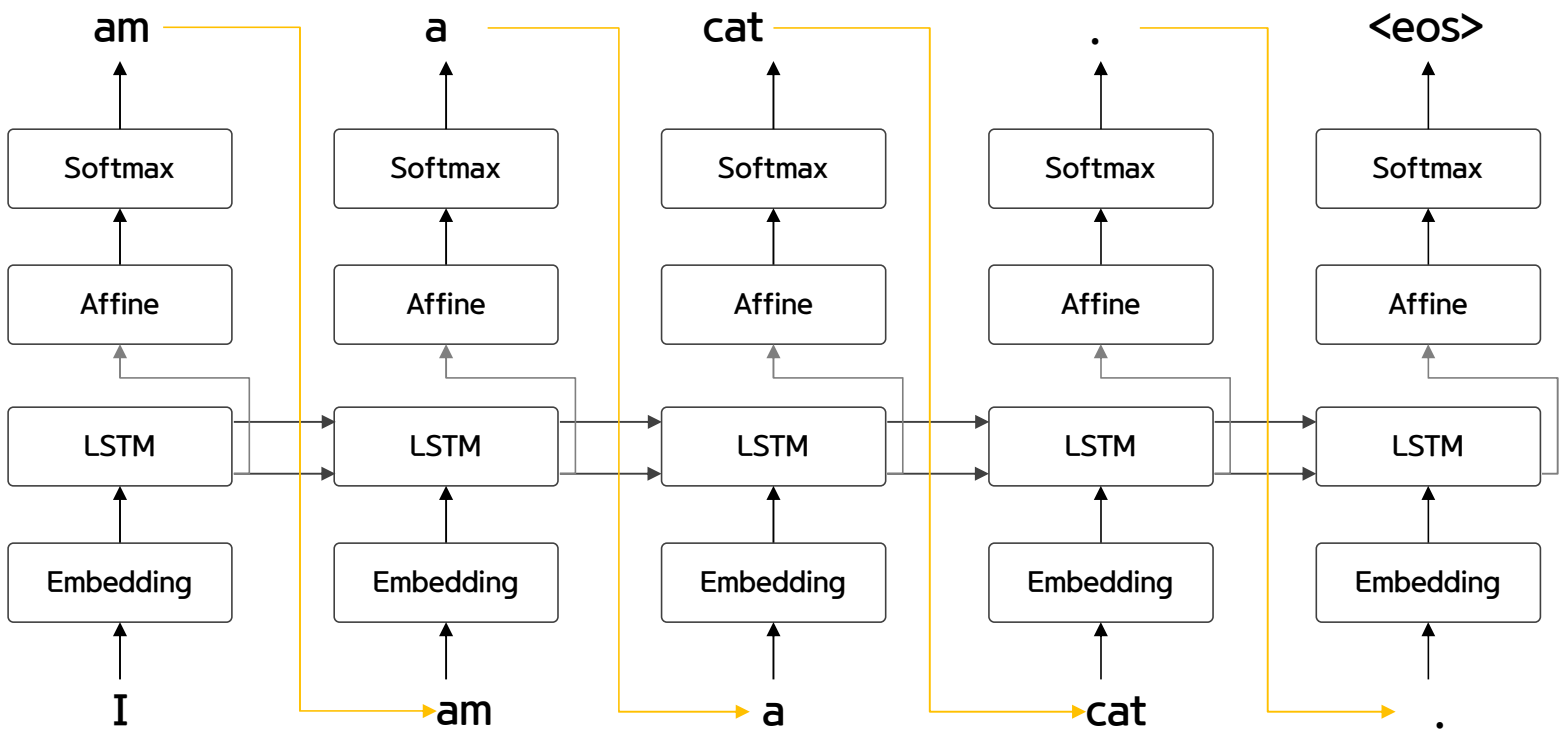
순전파 때는 Encoder 에서 인코딩된 정보가 LSTM 계층의 은닉 상태를 통해 Decoder 에 전해진다.

그리고 seq2seq 의 역전파 때는 이 가교를 통해 기울기가 Decoder 로부터 Encoder 로 전해진다.



텍스트 생성

텍스트 생성시 sequence의 입력과 출력



원핫 인코딩

원-핫 인코딩은 단어 집합의 크기를 벡터의 차원으로 하고, **표현하고 싶은 단어의 인덱스에 1의 값을 부여하고, 다른 인덱스에는 0을 부여**하는 단어의 벡터 표현 방식입니다. 이렇게 표현된 벡터를 **원-핫 벡터(One-Hot vector)**라고 합니다.

```
>>> a = tf.keras.utils.to_categorical([0, 1, 2, 3], num_classes=4)
>>> a = tf.constant(a, shape=[4, 4])
>>> print(a)
      tf.Tensor( [[1. 0. 0. 0.]    #  0 :  [ 1 0 0 0 ]
                  [0. 1. 0. 0.]    #  1 :  [ 0 1 0 0 ]
                  [0. 0. 1. 0.]    #  2 :  [ 0 0 1 0 ]
                  [0. 0. 0. 1.]],  #  3 :  [ 0 0 0 1 ]
                   shape=(4, 4), dtype=float32)
```


https://wikidocs.net/22647

텍스트 생성

- 입력문장

  “I've got a bad feeling about this”

- 출력문장

  I've got a bad feeling about this that doesnt like the kellswater irish of yore i love there there i gone away gone abusing love love mccorley goes to die on the bridge of toome today and see no hole i did stand i care right love hung love is sinking pair did love love love love love love me love seen town tree love love o ruler of love right right lovers i sinking marry me right right rigs right right sinking sinking irish love till high at sinking love there i love beside love love there for napper creole to and by night as i

The End