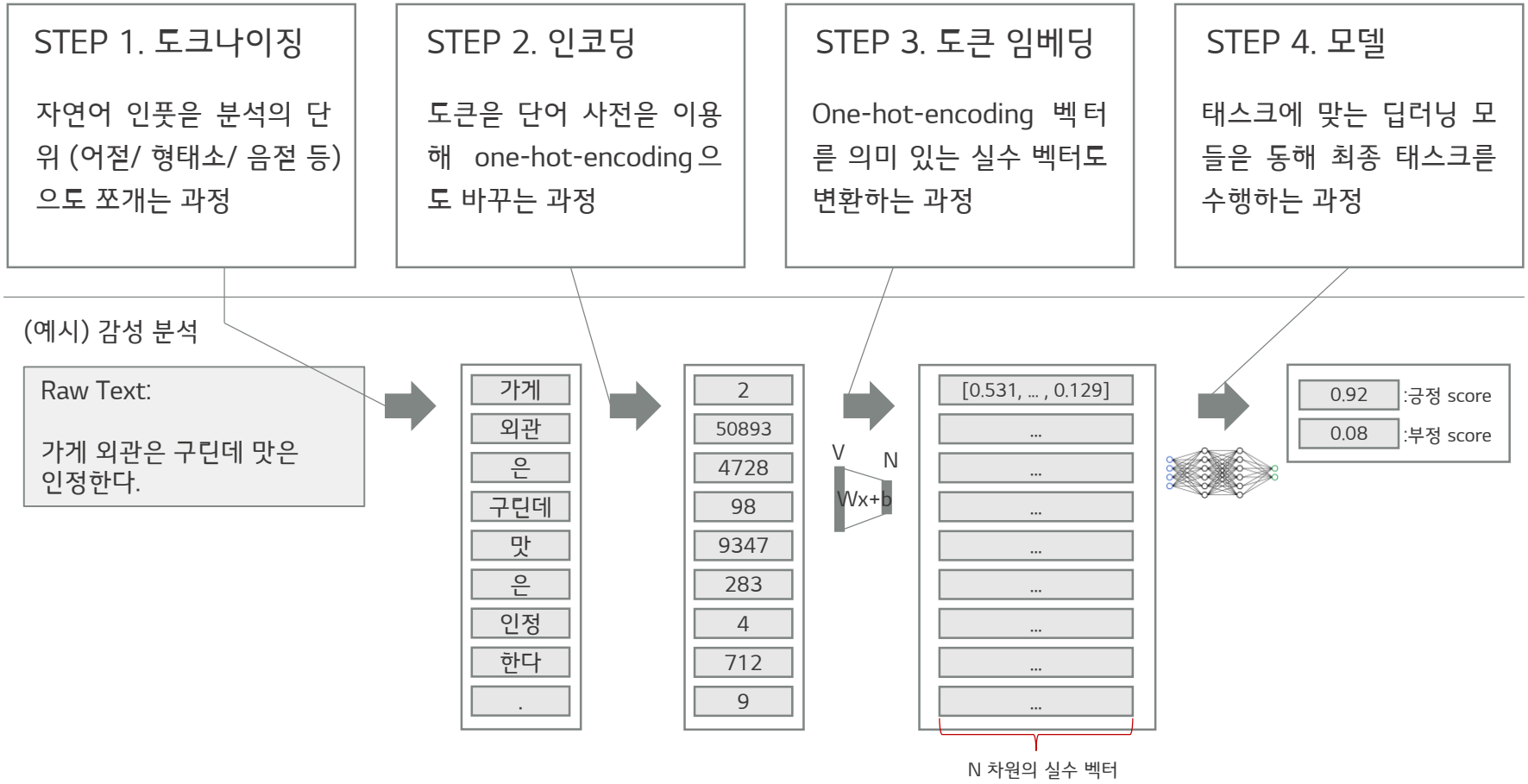
M3. NLU FLOW

일반적으로 NLU 태스크는 토크나이징 > 인코딩 > 토큰 임베딩 > 모델링 단계를 거친다.

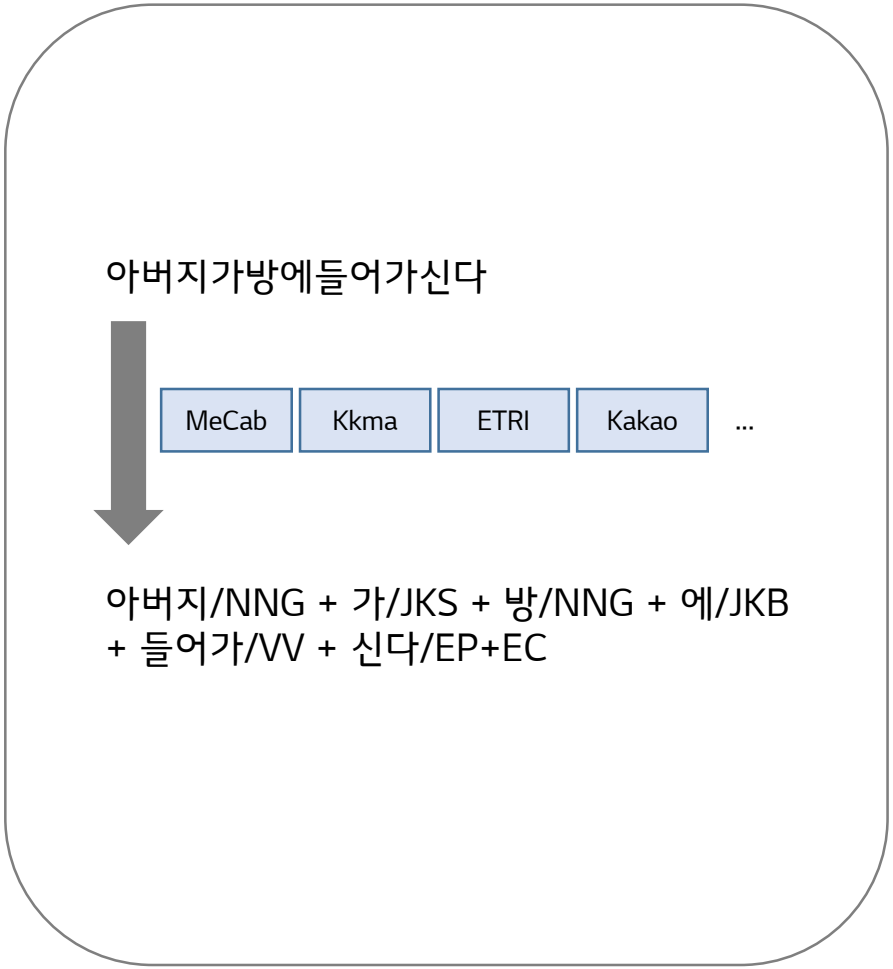


분석의 단위가 되는 어절 / 형태소 / 음절 / 자소

	문장	[자갈돌 깨뜨려 모래알]
	어절	[자갈돌] [깨뜨려] [모래알]
✓	형태소	[자갈] [돌] [깨뜨려] [모래] [알]
✓	음절	[자] [갈] [돌] [깨] [뜨] [려] [모] [래] [알]
	자소	ㅈ + ㅏ + ㄱ + ㅏ + ㄹ + ㄷ + ㅗ + ㄹ + ㄲ + ㅐ + ㄸ + ㅡ + …

- **영어**는 어절 단위로 분석하는 경우가 많다.
- 복합어인 **한국어**는 어절보다 작은 의미 단위로 쪼개야 성능이 좋다. -> 형태소 분석

한국어 형태소 분석



특징

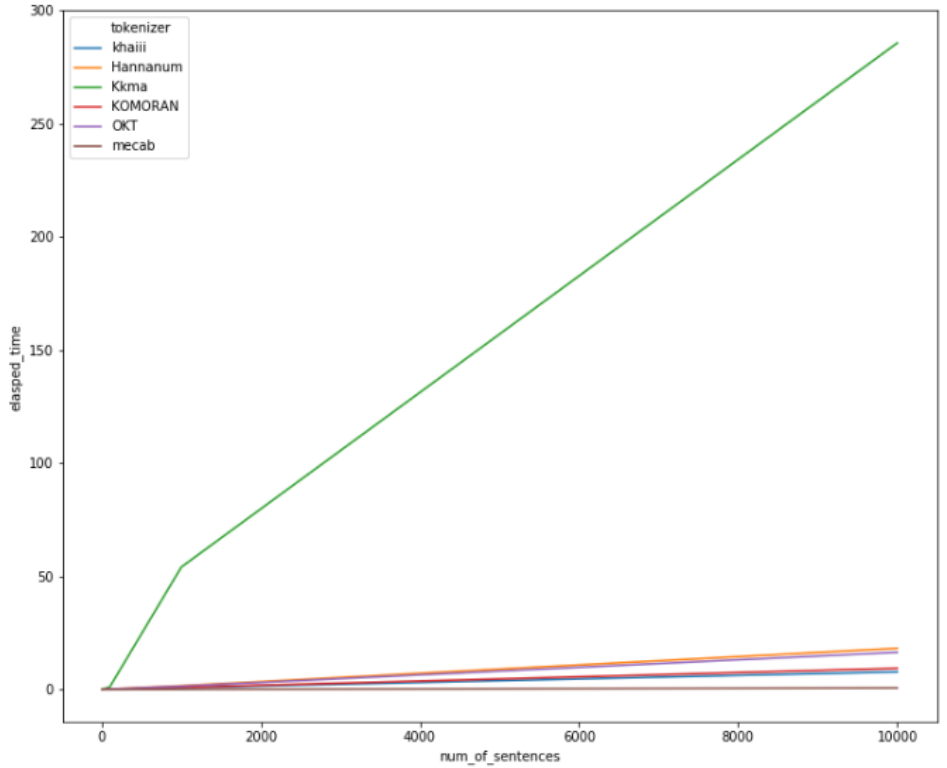
- 형태소 분석은 한국어 자연어처리의 기본.
- 사용할 데이터의 특성(띄어쓰기 유무 등)이나 개발 환경(Python, Java)에 따라서 적합한 형태소 분석기를 선택
- MeCab : 처리 속도가 빠르고 품질도 상위권.
- Khaiii : 카카오에서 만든 딥러닝 기반의 형태소 분석기

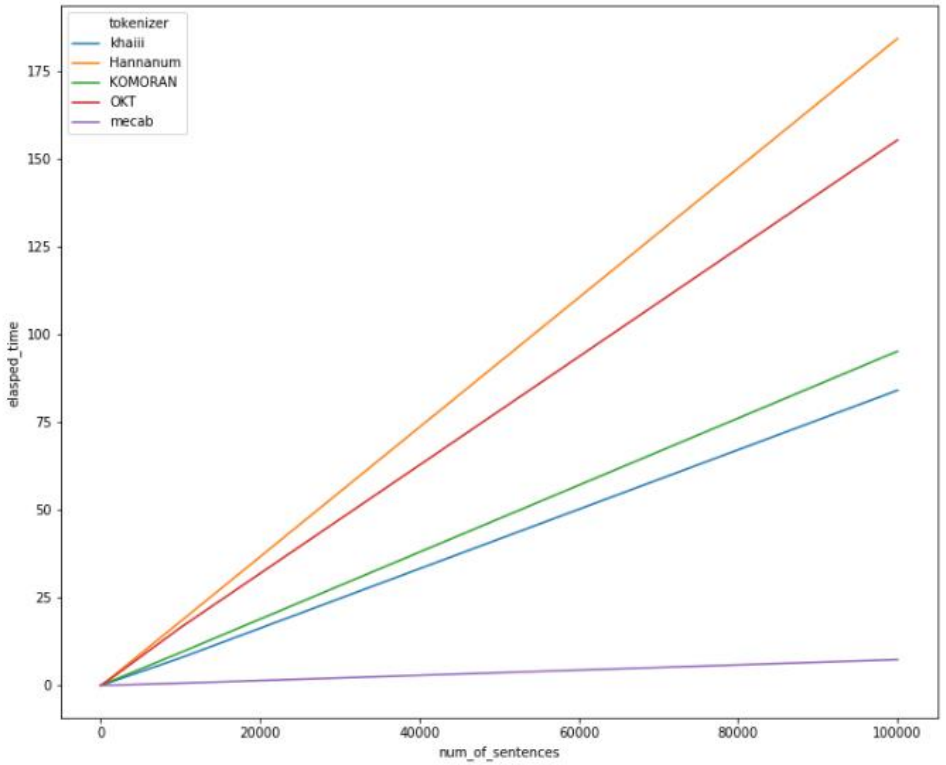
단점

- 분석기마다 품사 tag도 조금씩 다름. (동일X)
- 단어사전 생성 시 OOV 문제 (한국어 단어가 넘나 많다)

한국어 형태소 분석기 비교

- 한나눔, 꼬꼬마(kkma), 코모란, OKT 분석기는 konlpy(코엔엘파이) 설치로 이용 가능





- 꼬꼬마 분석기는 분석 속도 면에서 이슈 있음.
- MeCab 분석기의 속도가 월등히 빠른 편

한국어 형태소 분석기 비교

미니_실습1_한국어_형태소_분석.ipynb

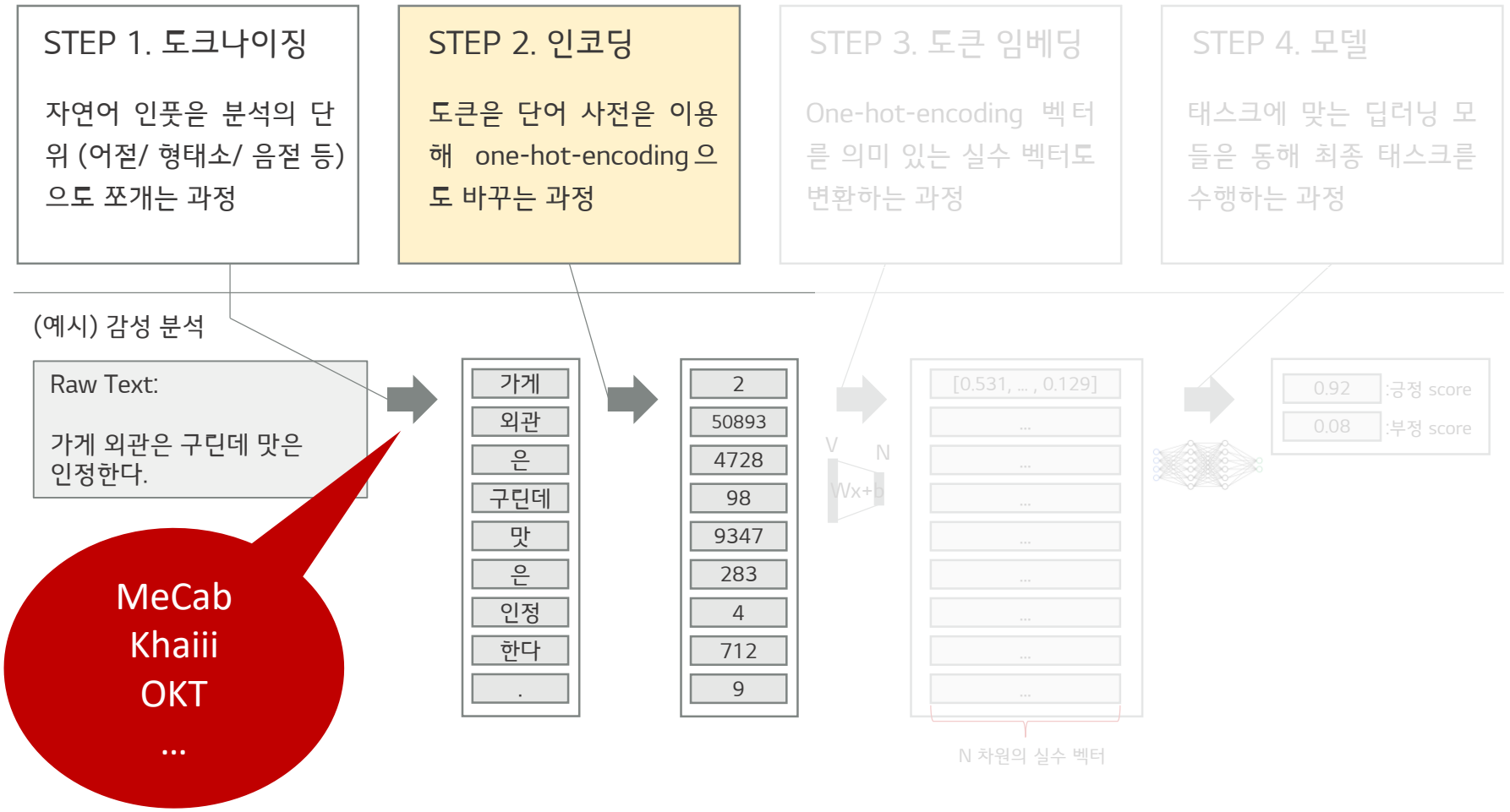
- 분석 결과 예시: “너무기대안하고갔나재밌게봤다”

https://iostream.tistory.com/144

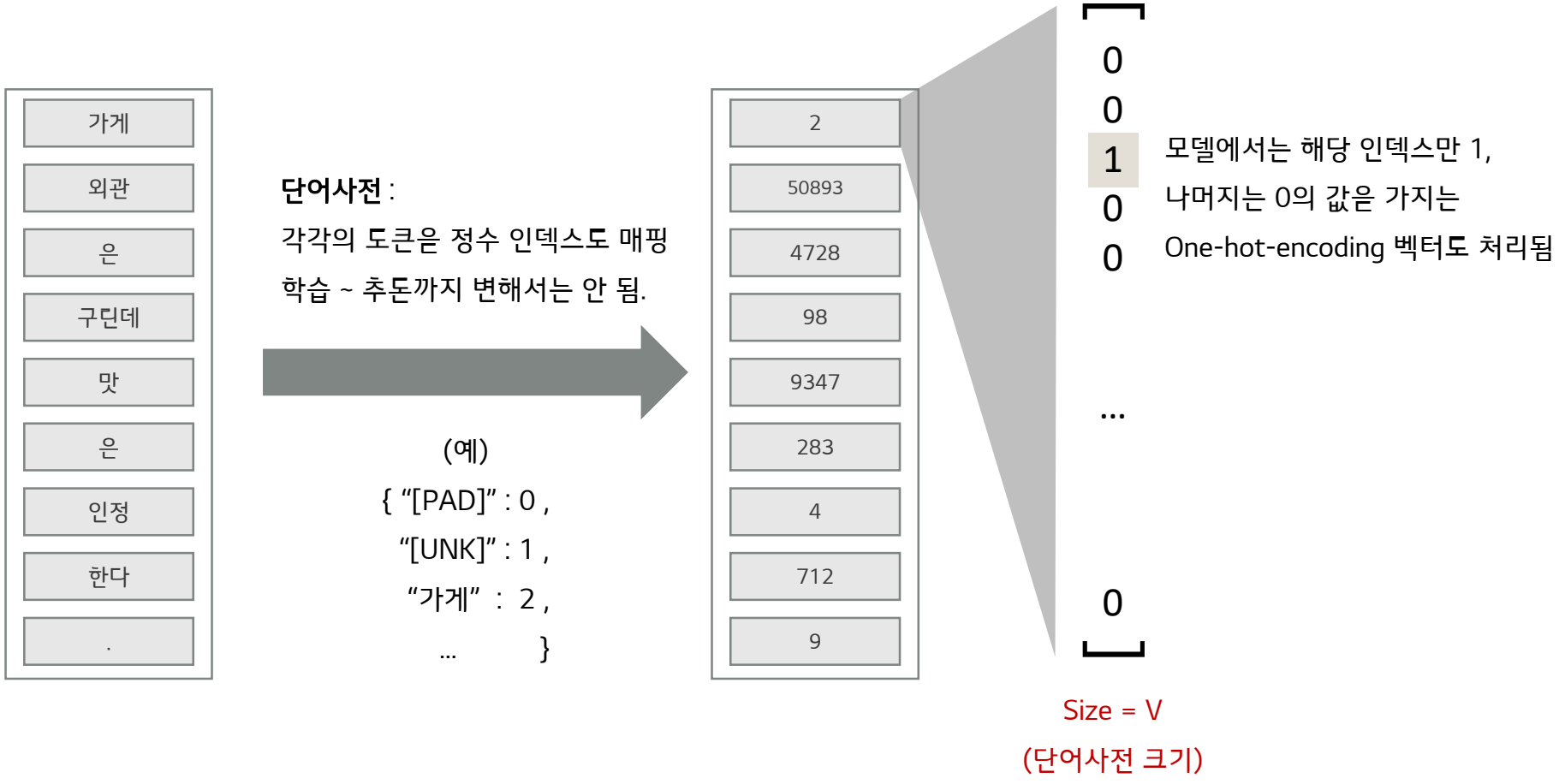
khaiii	한나눔	꼬꼬마	KOMORAN	OKT	MeCab ✓DAP-talk
너/MAG	너무기대안하고갔나재밌게봤다/N	너무/MAG	너무/MAG	너/Modifier	너무/MAG
무/NNG		기대/NNG	기대/NNG	무기/Noun	기대/NNG
기/MAG		안/NNG	안/NNG	대안/Noun	안/MAG
대안/NNG		하/XSV	하/XSV	하고/Josa	하/VV
하/XSV		고/ECE	고/EC	갔나/Verb	고/EC
고/EC		가/VV	가/VX	재밌게/Adjective	갔/VV+EP
가/VX		었/EPT	았/EP	봤다/Verb	나/EC
았/EP		나/EFQ	나/EC		재밌/VA
나/VV		재밌/VA	재밌/VA		게/EC
재밌/VA		게/ECD	게/EC		봤/VX+EP
게/EC		보/VXV	보/VX		다/EC
보/VV		았/EPT	았/EP		
았/EP		다/EFN	다/EC		
다/EC					

원형 보존

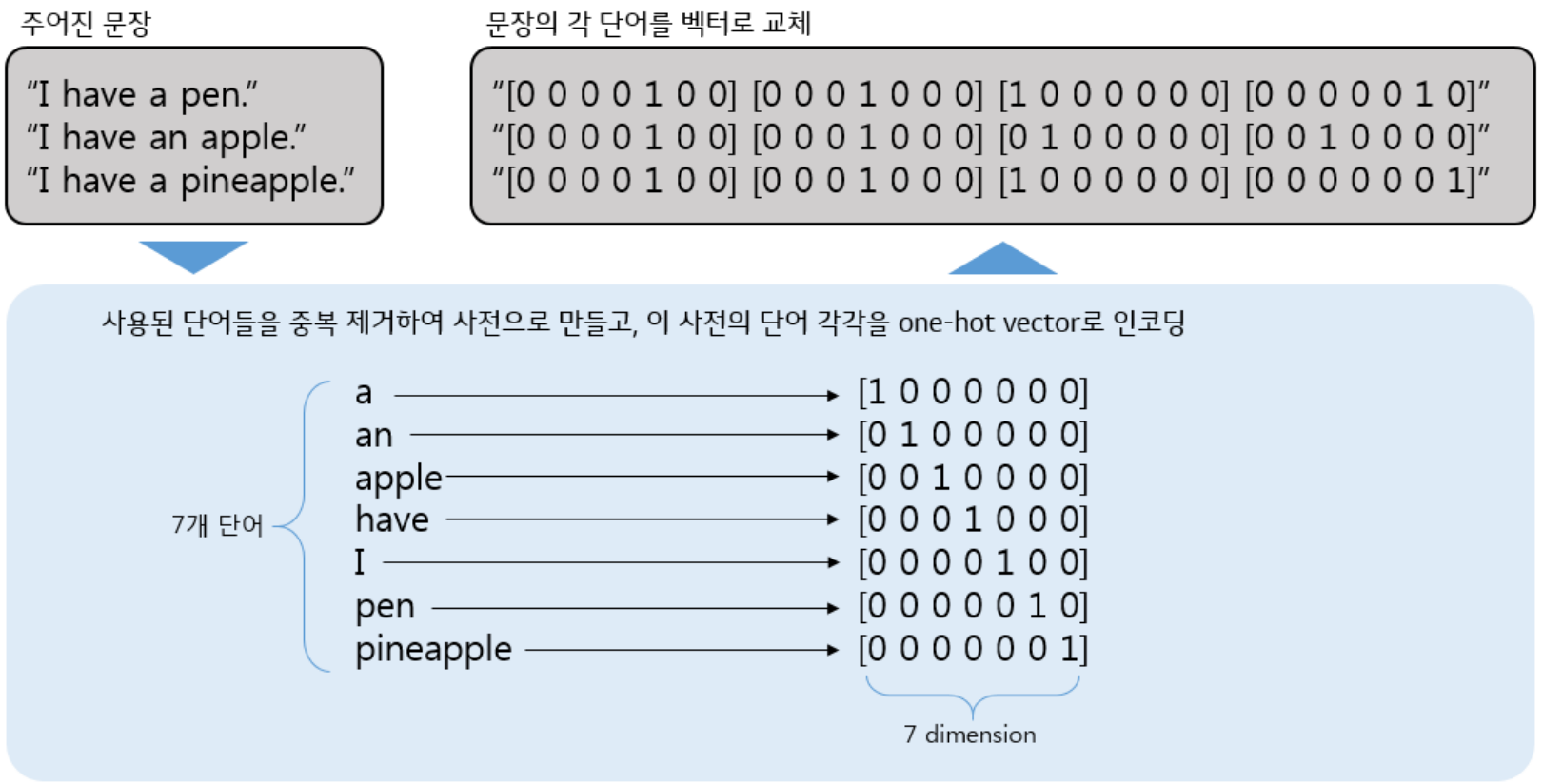
자연어도 되어 있는 '도큰'은 기계에 인식시키기 위해서는 인코딩 단계를 거쳐야 한다.



토큰을 단어 사전을 이용해 one-hot-encoding으로 바꾸는 과정



(참고) one-hot encoding



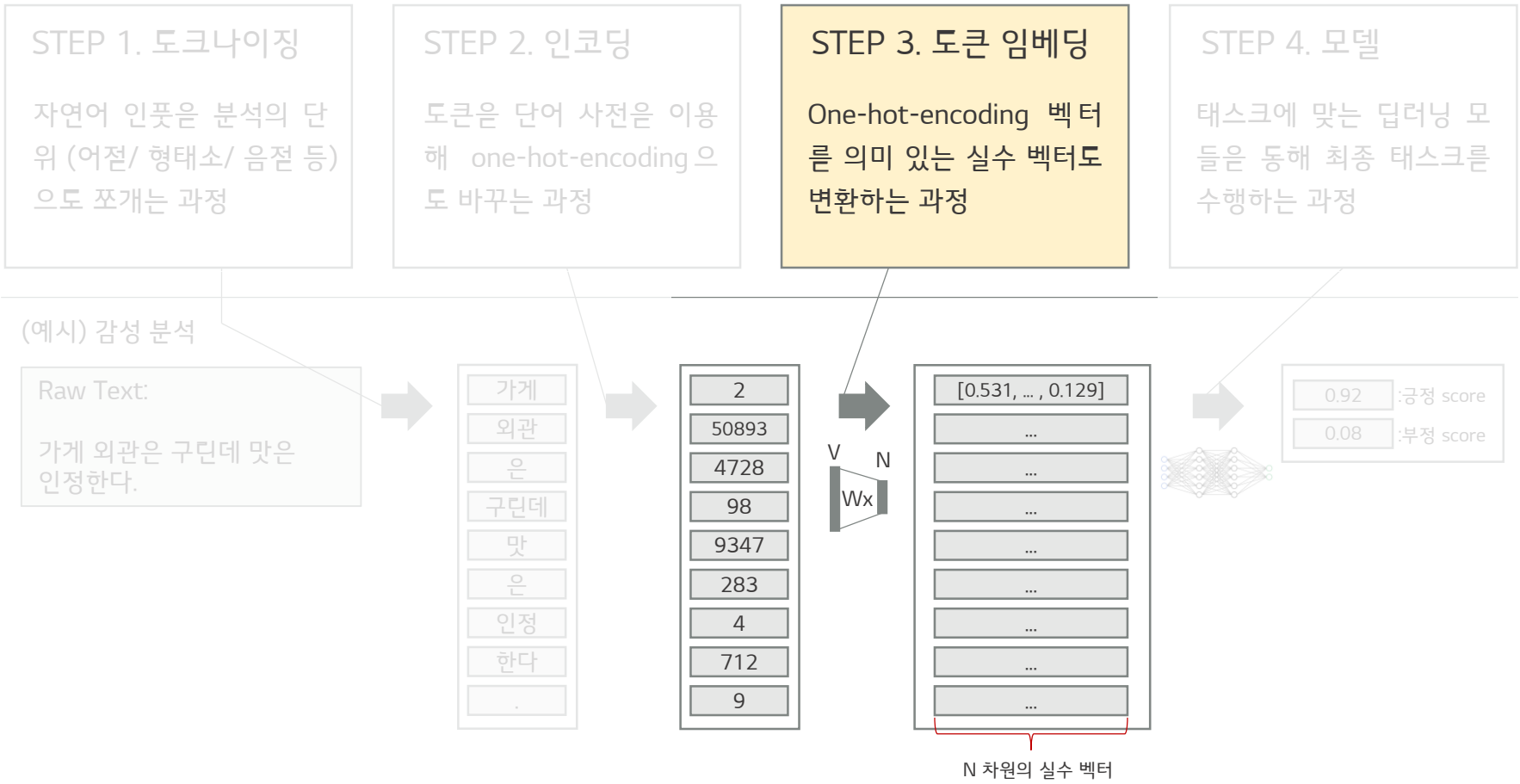
단어사전 만들기

- 가지고 있는 코퍼스에 대해 중복을 제거하고 빈도수 기준 Top V개를 이용해 사전을 만든다.
- (예) 위키 백과, 뉴스를 크롤링해서 형태소분석을 시행한 후 상위 빈도 10만개의 단어 사용

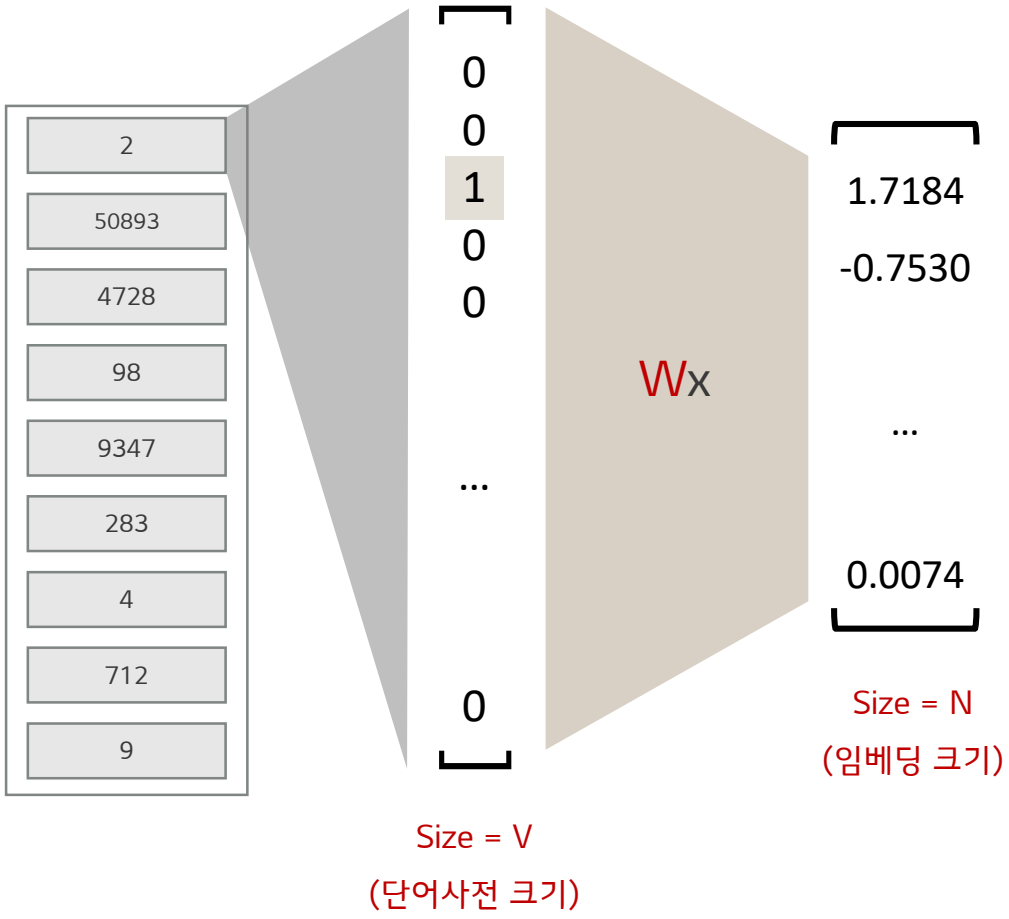
단어	빈도	
가	1,100,194	빈도수 Top-10만개 단어 -> 단어사전에 포함
은	1,099,901	
...	...	
뱕다	9	사전에 포함되지 않은 단어는 [UNK] 처리 됨
냔냐	2	

- 사전에 포함되지 않은 단어를 처리하기 위해 [UNK] 도큰을 사전에 반드시 추가
- 딥러닝 배치 처리를 위해 시권스 길이를 맞추기 위해 [PAD] 도큰도 사전에 추가

- 이외에 번역과제를 위해 문장의 시작을 알리는 <sos>, 끝을 알리는 <eos> 등 스페셜 도큰을 사전에 포함할 수 있음.


STEP 1. 도크나이징
자연어 인풋은 분석의 단위 (어절/ 형태소/ 음절 등)으로 쪼개는 과정
STEP 2. 인코딩
도큰은 단어 사전을 이용해 one-hot-encoding으로 바꾸는 과정
STEP 3. 도큰 임베딩
One-hot-encoding 벡터를 의미 있는 실수 벡터로 변환하는 과정
STEP 4. 모델
태스크에 맞는 딥러닝 모들을 동해 최종 태스크를 수행하는 과정
(예시) 감성 분석
Raw Text:
가게 외관은 구딘데 맛은 인정한다.
가게
외관
은
구딘데
맛
은
인정
한다
.
2
50893
4728
98
9347
283
4
712
9
V
N
Wx
[0.531, ... , 0.129]
...
...
...
...
...
...
...
...
N 차원의 실수 벡터
0.92
:긍정 score
0.08
:부정 score

One-hot-encoding된 벡터는 적당한 차원의 실수 벡터로 임베딩 한다.



QUIZ

인풋 인코딩이 (Vx1)차원 벡터일 때
W의 차원은 ?

A

W : N x V 차원

- 단어사전의 크기 (V)가 커지면 임베딩에 필요한 파라메터 수가 급증한다!
- 예를 들어 V=10만, N=300이면 W는 3000만 차원
- 반대로 단어사전 크기가 너무 작으면 [UNK] 개수가 늘어나기 때문에 적정 개수를 유지하는 것이 중요함.

(참고) one-hot vector에 대한 임베딩은 Look-up table과 같이 계산할 수 있음

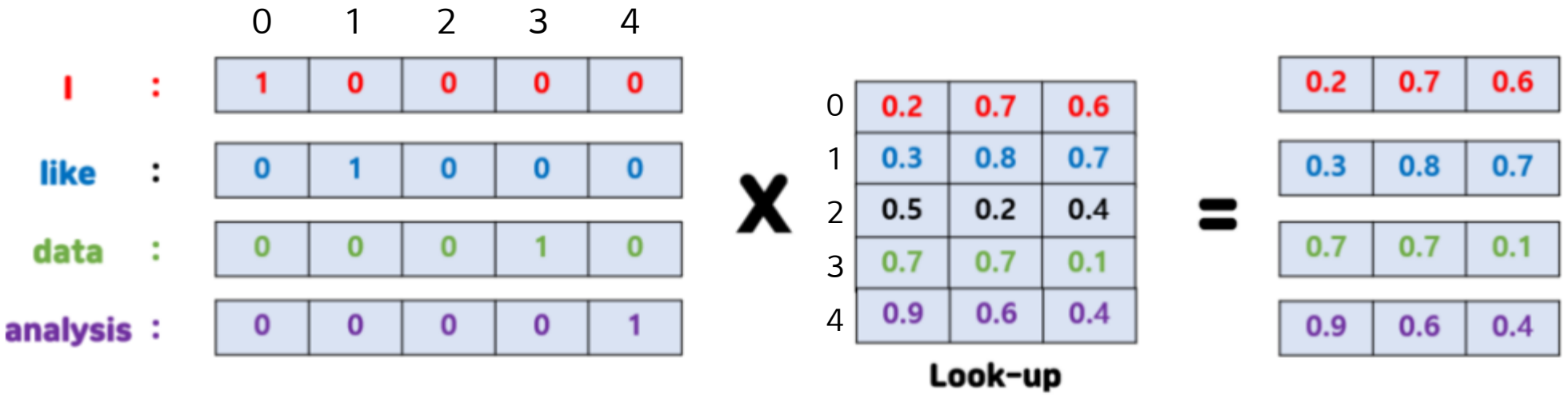
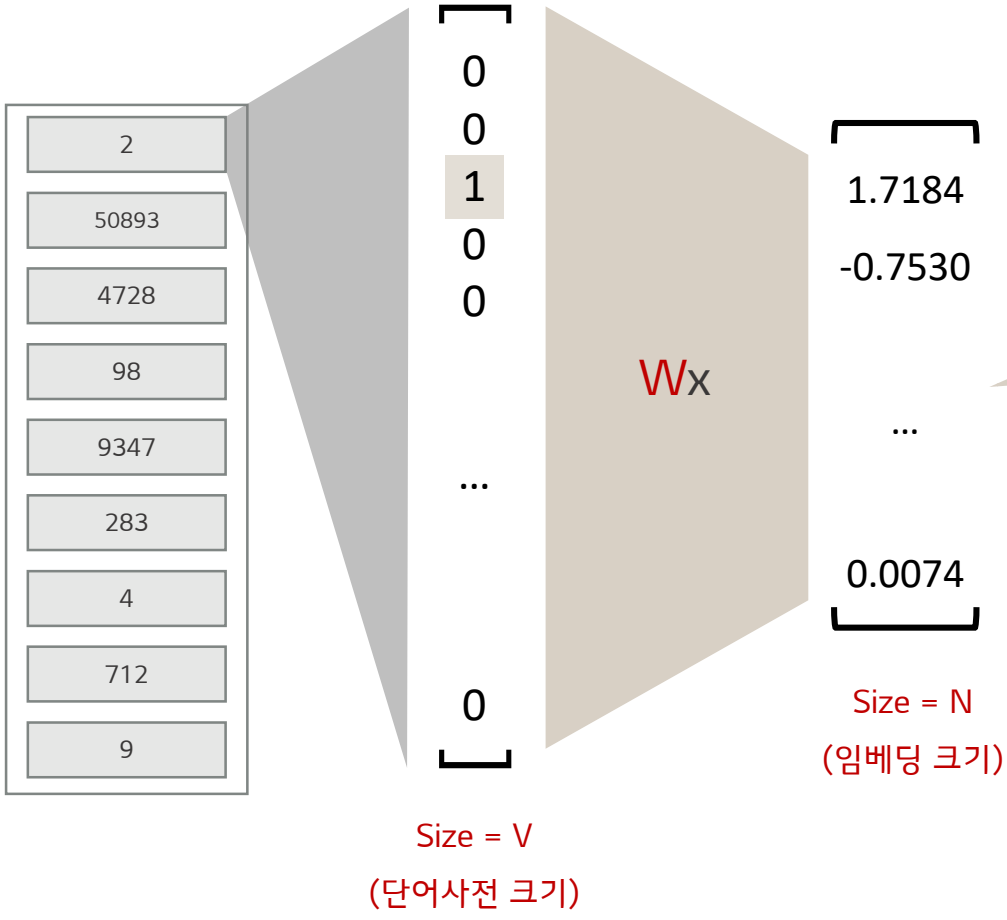


그림 출처: https://blog.naver.com/ollehw/221749978611

단어를 의미 있는 벡터로 임베딩 하는 방법 ?



임베딩 된 벡터가 의미 있으면 더 좋지 않을까?

'의미 있는' 벡터?

- 비슷한 의미를 가지는 단어를 비슷한 공간으로 임베딩 한다.
- (예) E(초콜릿) ≈ E(초코)

사람은 단어의 의미를 어떻게 배우는가

-> 단어의 의미는 해당 단어의 문맥(context)이 담고 있다.

빈 칸에 알맞은 말은?

- 친구의 자취방은 생각보다 ____했다.
- 새도 산 면도기가 좋은지 아주 면도가 ____하게 되었다.
- 야, 화장실좀 제발 ____하게 쓰자!

같은 문맥에서, 동일한 위치에 등장할 수 있는 단어들은 유사한 의미를 가졌을 것이다 !!

'매나니' 라는 단어를 아시나요?

- 삽이라도 있어야 땅을 파지 *매나니*도야 어떻게 하겠나?
- 그 녀석 무슨 배짱인지 *매나니*로 와서 일을 하겠다고 한다.
- 김치도 없이 *매나니*도 어떻게 끼니를 때웁니까, 참치캔이라도 따서 드세요.

단어의 사전적 정의를 몰라도,
등장하는 문맥을 여덟 보면
단어의 의미 특징을 뽑아낼 수 있다 !!

① CBOW : Continuous Bag of Words

- 수많은 코퍼스를 이용해서 단어 임베딩을 만드는 방법을 학습시키는 unsupervised learning 방법
- IDEA :

✓ 친구의 자취방은 생각보다 [ ] 했다.
✓ 새로 산 면도기가 좋은지 아주 면도가 [ ]하게 되었다.
✓ 야, 화장실 좀 제발 [ ] 하게 쓰자!

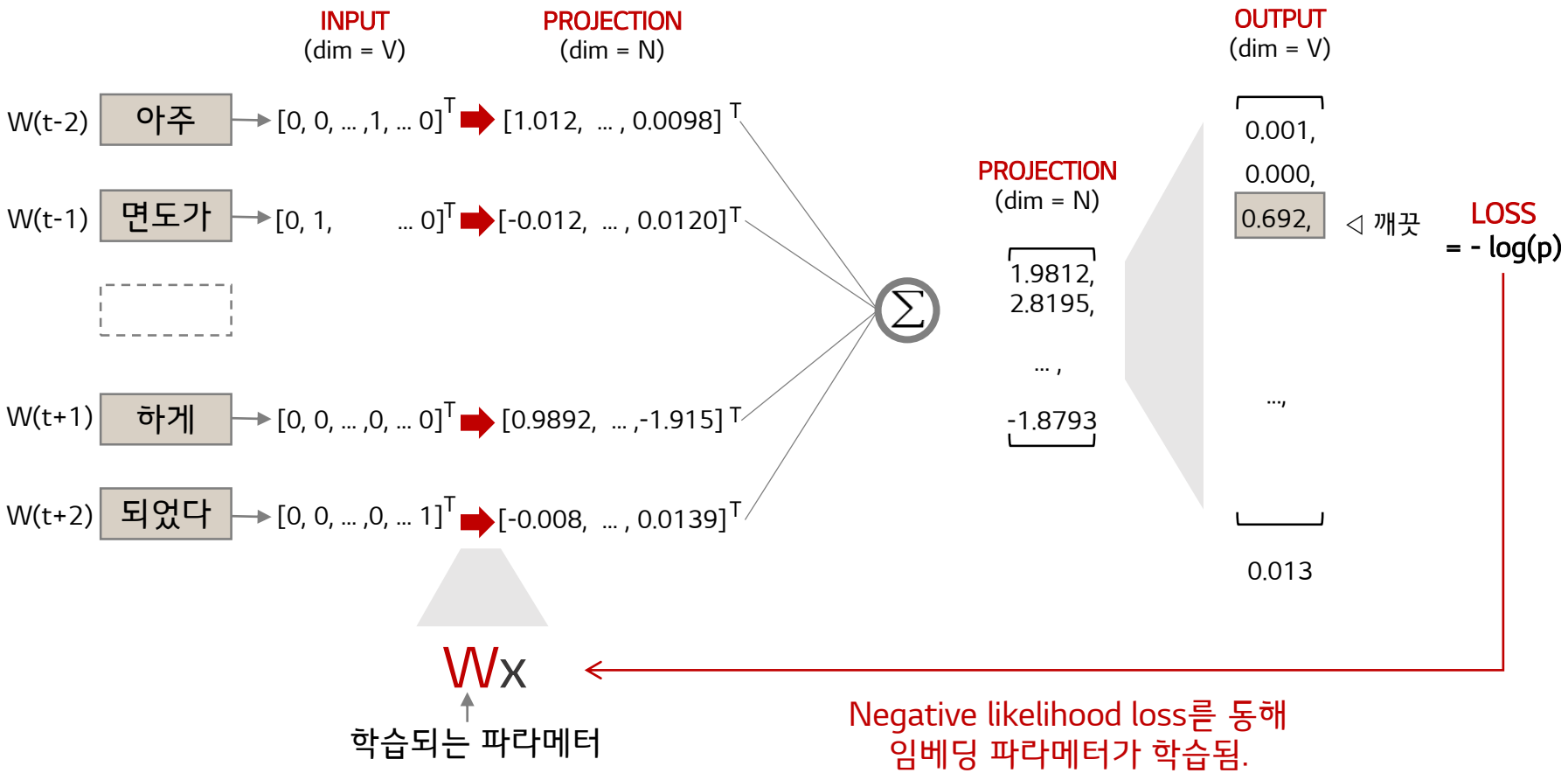
[ ] 안에 들어갈 단어는 무엇일까요?

모델이 문서들을 읽으며 '빈칸 맞추기' 태스크를 수행하게 하자

직장인 곽모(32)씨는 지난 ? 부터 게임기 '닌텐도 스위치'를 구하지 못해 고심이다. 지난해말 30만원대 초반이던 온라인 가격이 최근 60만원대로 폭등한 탓이다. 곽씨는 정가 구매를 위해 이른 아침 근처 대형 마트를 찾았지만, 대기자가 10명을 넘어 이틀 연속 구매에 실패했다. 곽씨는 "1월엔 어디서나 쉽게 구할 수 있던 게임기가 품절되니 당황스럽다"며 "어서 '모여봐요 동물의 숲'을 해보고 싶은데 웃돈을 주고 구매해야 하나 고민"이라고 말했다.

관련 논문: Efficient estimation of word representations in vector space (ICLR 2013)

① CBOW : Continuous Bag of Words



① CBOW step by step

Data 준비

Raw text

아프리카 평원에 동이 트며 프라이드 랜드의 모든 동물들이 프라이드 닥 앞으도 모여든다. 라피키는 아기 사자 심바를 프라이드 닥 위에서 동물들에게 들어보이며 새도운 후계자의 탄생을 알린다. ...

토크나이즈

['아프리카', '평원', '에', '동', '이', '트', '며', '프라이드', '랜드', '의', '모든', '동물', '들', '이', '프라이드', '닥', '앞', '으도', '모여', '든다', '.', '라피', '키', '는', '아기', '사자', '심바', '를', '프라이드', '닥', '위', '에서', '동물', '들', '에게', '들', '어', '보이', '며', '새도운', '후계자', '의', '탄생', '을', '알린다', '.', ...]

학습 가능한 형태로 변환

(예) window size = 2

X	Y
아프리카, 평원 , 동 , 이	에
평원, 에, 이, 트	동
에, 동, 트, 며	이
동, 이, 며, 프라이드	트
...	...

① CBOW step by step

인풋 projection

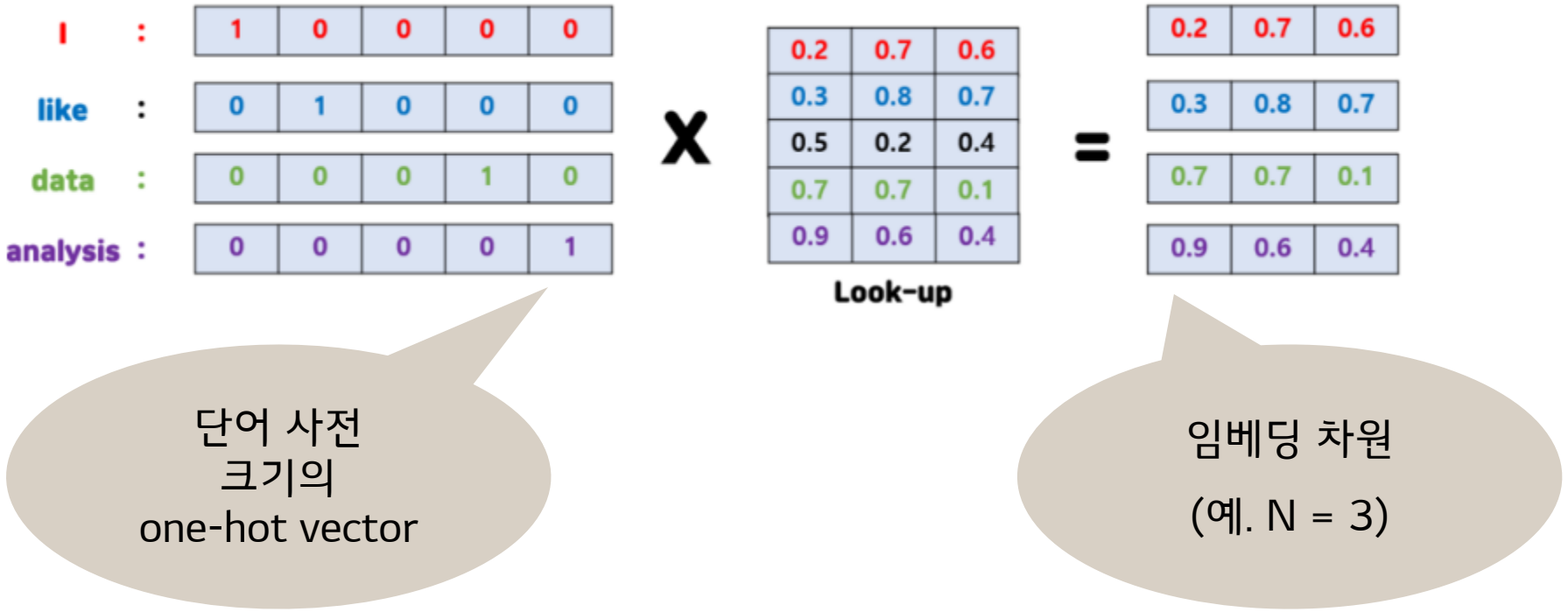
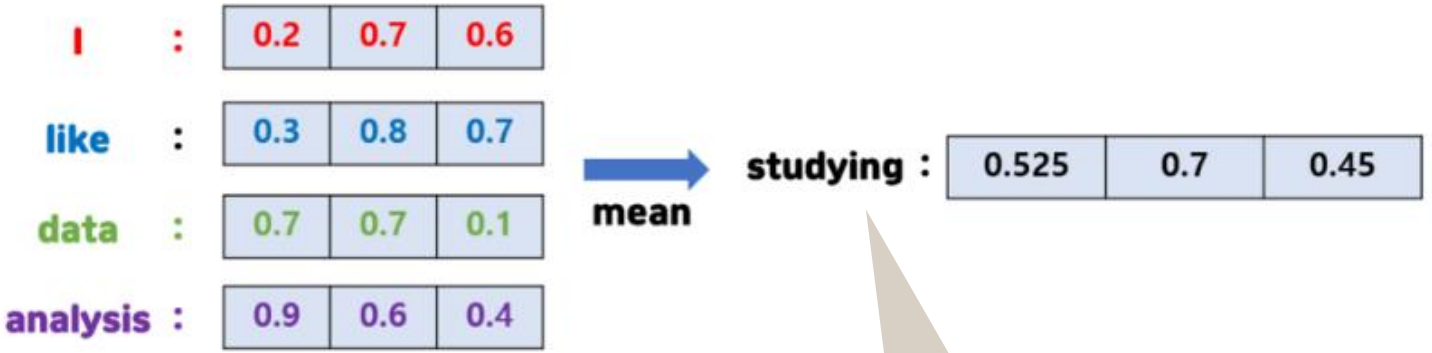


그림 출처: https://blog.naver.com/ollehw/221749978611

① CBOW step by step

Projection된 벡터를 하나의 벡터로 나타냄



가방 안에 들어있는 단어들의 정보를 통해

가운데 들어갈 단어의 정보를 유추

그림 출처: https://blog.naver.com/ollehw/221749978611

① CBOW step by step

추춘된 정보가 정말 가운데에 들어갈 단어를 나타내는지 학습

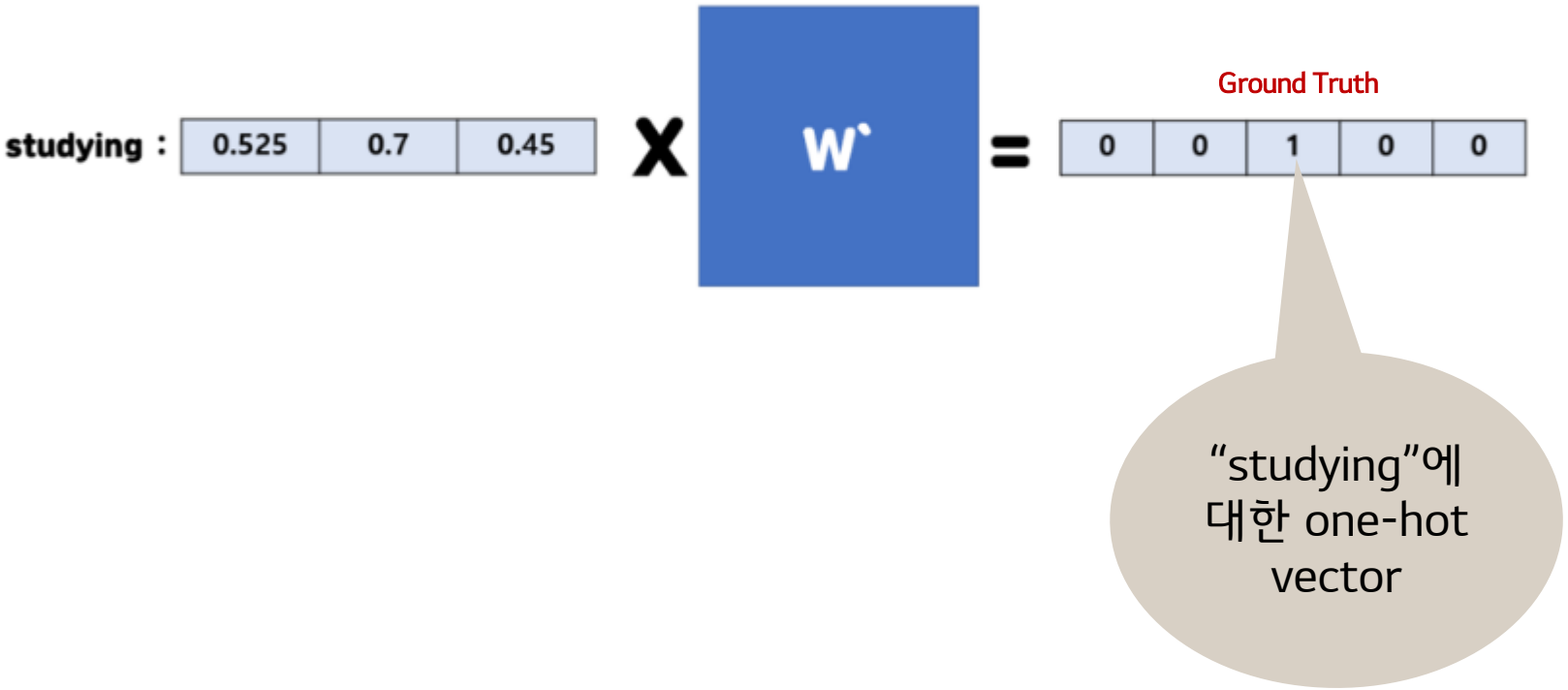


그림 출처: https://blog.naver.com/ollehw/221749978611

① CBOW step by step

추춘된 정보가 정말 가운데에 들어갈 단어를 나타내는지 학습

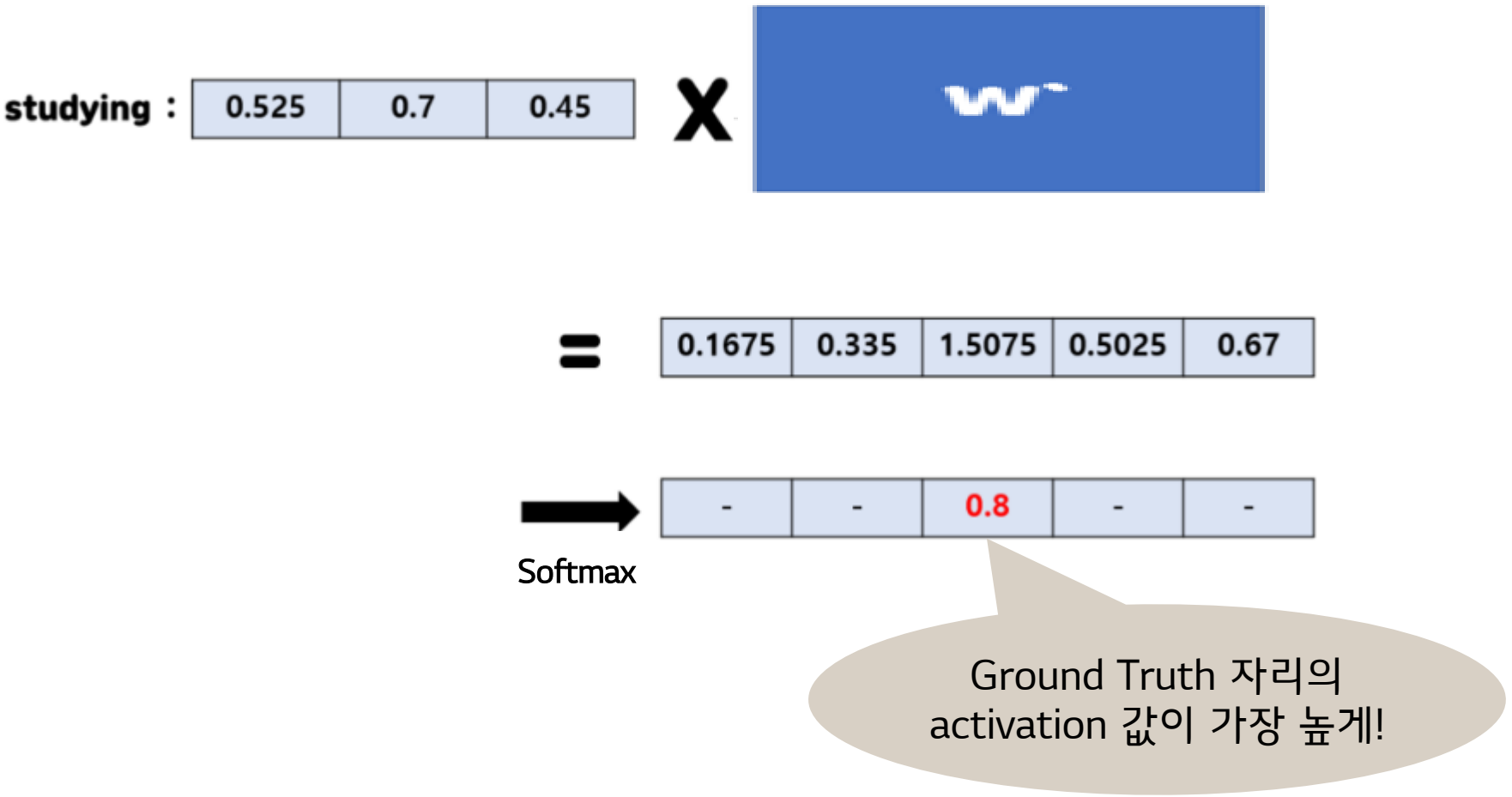
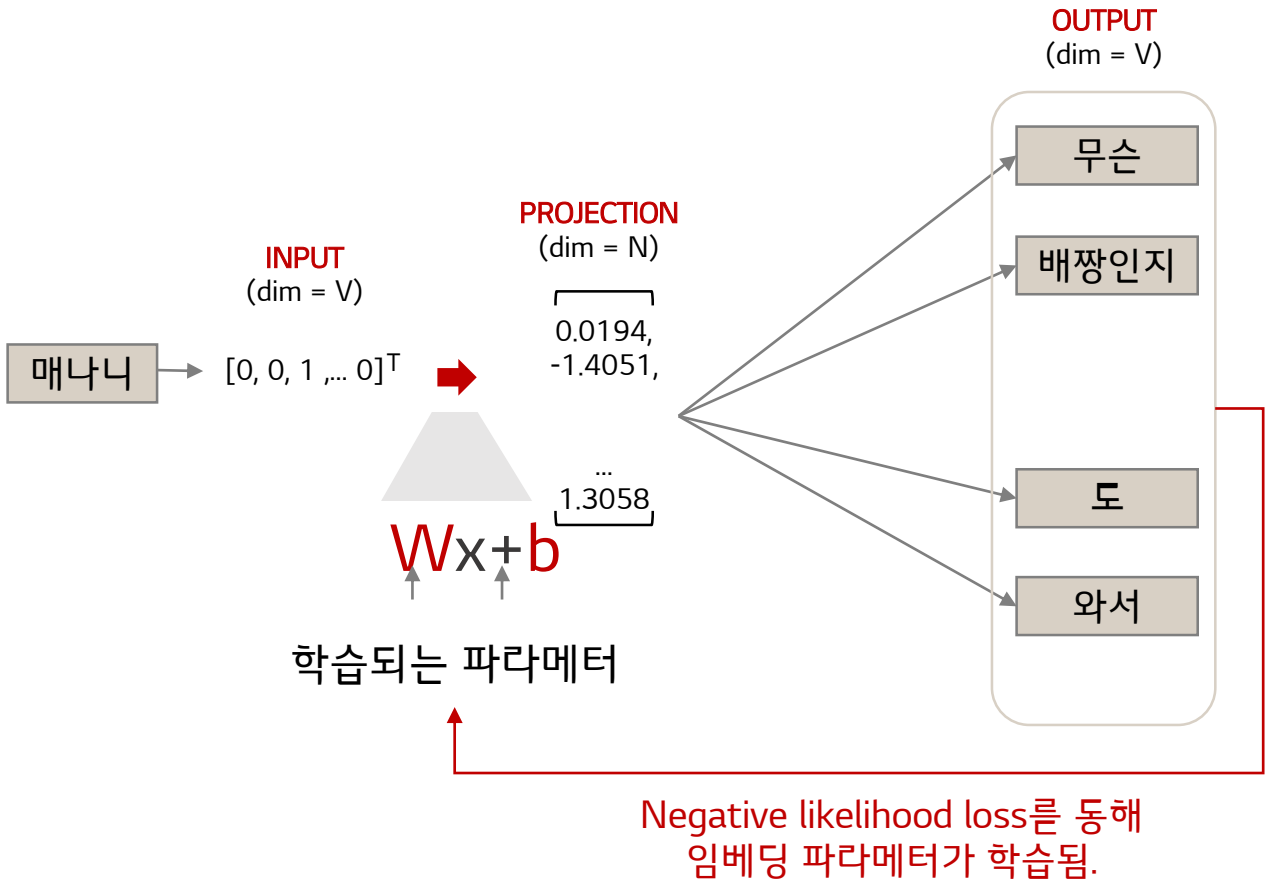


그림 출처: https://blog.naver.com/ollehw/221749978611

② Skip-gram

- CBOW가 중간에 있는 단어를 맞추는 것이었다면, Skip-gram은 단어 주변에 오는 단어를 맞추는 식으로 학습

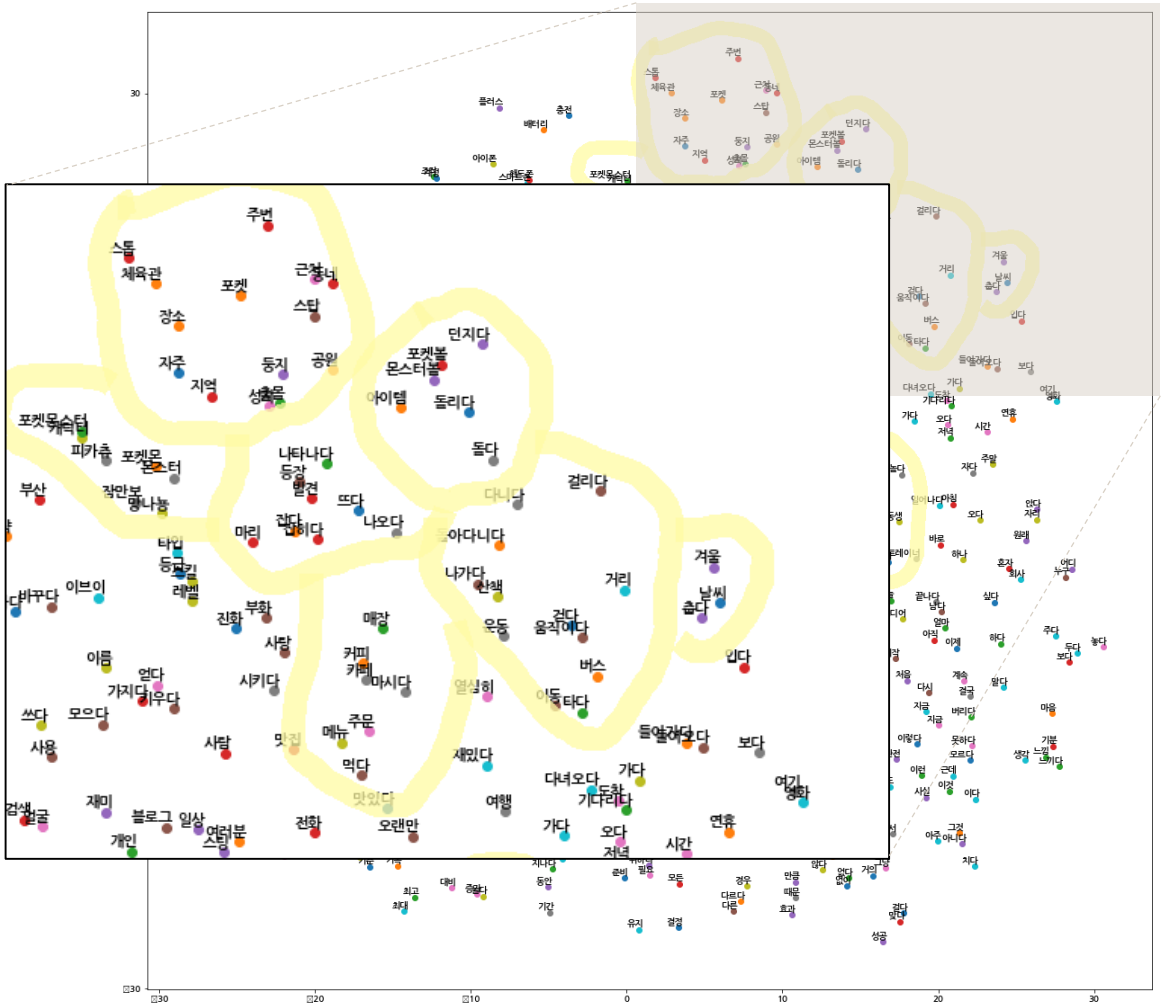


CBOW, Skip-gram으로 학습된 워드 임베딩이 가지는 특징

저장공간의 효율적 사용

유사한 단어 벡터의 밀집

의미를 담은 수치연산 가능



의미를 담은 수치연산 예시

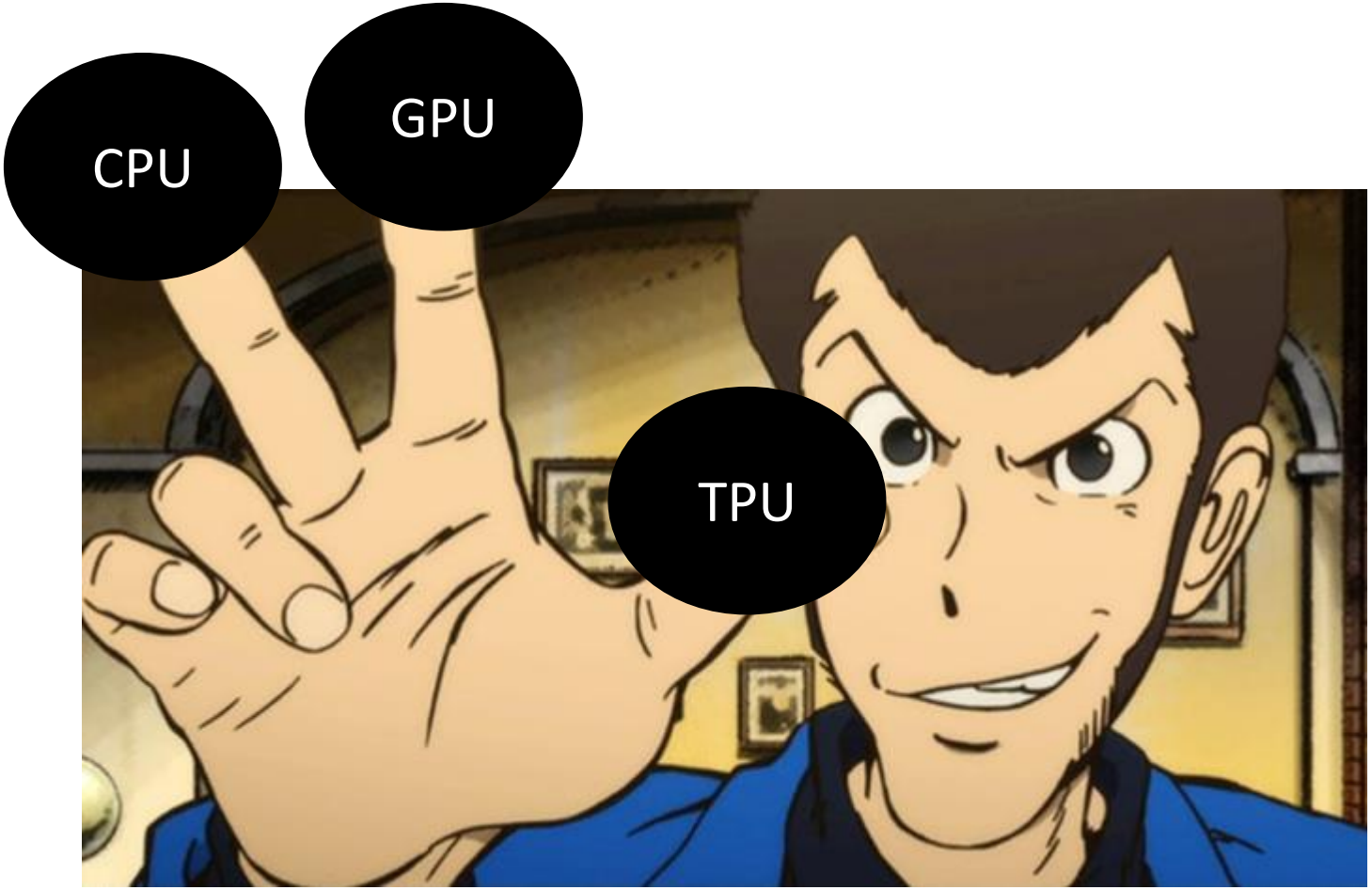
Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

관련 논문: Efficient estimation of word representations in vector space (ICLR 2013)

실습 준비 – Colab 사용하기

- 클라우드에서 머신러닝을 실행할 수 있는 주피터 노트북 환경
- TensorFlow, scikit-learn, numpy 등 머신러닝에 필요한 라이브러리 대부분이 미리 셋팅되어 있음.



실습 준비 – Colab 사용하기

실습 준비 – TensorFlow 2.0

TensorFlow 1.x

```
import tensorflow as tf

## 그래프를 정의합니다
g = tf.Graph()
with g.as_default():
    x = tf.placeholder(dtype=tf.float32,
                       shape=(None), name='x')
    w = tf.Variable(2.0, name='weight')
    b = tf.Variable(0.7, name='bias')
    z = w * x + b
    init = tf.global_variables_initializer()

## 세션을 만들고 그래프 g를 전달합니다
with tf.Session(graph=g) as sess:
    ## w와 b를 초기화합니다
    sess.run(init)
    ## z를 평가합니다
    for t in [1.0, 0.6, -1.8]:
        print('x=%4.1f --> z=%4.1f'%(
              t, sess.run(z, feed_dict={x:t})))
```

Wx + b 계산하는 것도

왜 이렇게 복잡해

실습 준비 – TensorFlow 2.0

TensorFlow 1.x

```
import tensorflow as tf

## 그래프를 정의합니다
g = tf.Graph()
with g.as_default():
    x = tf.placeholder(dtype=tf.float32,
                       shape=(None), name='x')
    w = tf.Variable(2.0, name='weight')
    b = tf.Variable(0.7, name='bias')
    z = w * x + b
    init = tf.global_variables_initializer()

## 세션을 만들고 그래프 g를 전달합니다
with tf.Session(graph=g) as sess:
    ## w와 b를 초기화합니다
    sess.run(init)
    ## z를 평가합니다
    for t in [1.0, 0.6, -1.8]:
        print('x=%4.1f --> z=%4.1f'%(
              t, sess.run(z, feed_dict={x:t})))
```

TensorFlow 2.0

```
import tensorflow as tf

w = tf.Variable(2.0, name='weight')
b = tf.Variable(0.7, name='bias')

# z를 평가합니다
for x in [1.0, 0.6, -1.8]:
    z = w * x + b
    print('x=%4.1f --> z=%4.1f'%(x, z))
```

실습 준비 – TensorFlow 2.0

TF Keras 빠르게 배우기 -> https://www.tensorflow.org/guide/keras/overview?hl=ko

```
from tensorflow import tf

model = tf.keras.Sequential()
# 64개의 유닛을 가진 완전 연결 층을 모델에 추가합니다:
model.add(tf.keras.layers.Dense(64, activation='relu'))
# 또 하나를 추가합니다:
model.add(tf.keras.layers.Dense(64, activation='relu'))
# 10개의 출력 유닛을 가진 소프트맥스 층을 추가합니다:
model.add(tf.keras.layers.Dense(10, activation='softmax'))

# 컴파일
model.compile(optimizer=tf.keras.optimizers.Adam(0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 모델 훈련
model.fit(train_data, labels, epochs=10, batch_size=32)
# 모델 평가
model.evaluate(test_data, labels)
# 샘플 예측
```

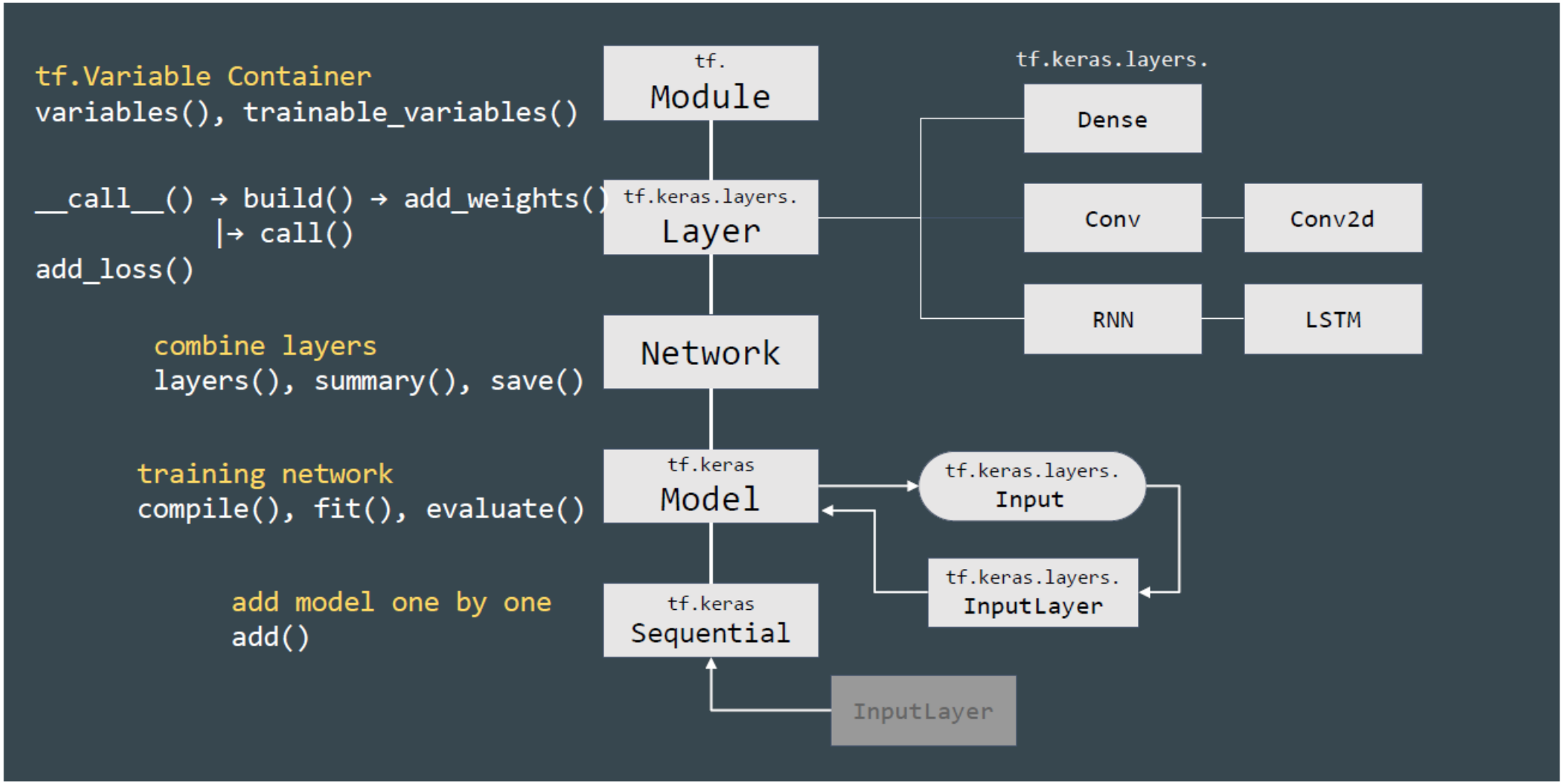
모델 구조 선언

컴파일

훈련 평가

예측 까지

실습 준비 – TensorFlow 2.0



TensorFlow 2.0 구현 연습 ☺

미니 실습2-TensorFlow2.0 연습.ipynb

Y = ax + b 구현하는 세 가지 방법

tf.keras.Sequential()

```
# Step1. 레이어를 쌓아 모델 구조 만들기
model = tf.keras.Sequential()
model.add(layers.Dense(1))

# Step2. 모델 컴파일
# -> loss, optimizer, metrics 지정
from tf.keras.optimizers import Adam
model.compile(optimizer=Adam(0.1),
              loss='mse')

# Step3. 모델 fitting
model.fit(X, Y, epochs = 10)
```

tf.keras.Model()

```
# Step1. 인풋 레이어 ~ 아웃풋까지 모델 연결
inputs = tf.keras.layers.Input(shape=(1,))
outputs = tf.keras.layers.Dense(1)(inputs)

# Step2. tf.keras.Model()을 통해
#        input과 output 지정
model = tf.keras.Model(inputs = inputs,
                       outputs = outputs)

# Step3. 모델 컴파일
# -> loss, optimizer, metrics 지정

model.compile(optimizer=Adam(0.1),
              loss='mse')

# Step4. 모델 fitting
model.fit(X, Y, epochs = 10)
```

Model Subclassing

```
# Step1. subclassing을 통해 모델 구현

class MyModel(tf.keras.Model):

  def __init__(self):
    super(MyModel, self).__init__()
    self.dense1 = tf.keras.layers.Dense(1)

  def call(self, inputs):
    outputs = self.dense1(inputs)
    return outputs

model = MyModel()

# Step2. 모델 컴파일
# -> loss, optimizer, metrics 지정
from tf.keras.optimizers import Adam
model.compile(optimizer=Adam(0.1),
              loss='mse')

# Step3. 모델 fitting
model.fit(X, Y, epochs = 10)
```

실습 1. 단어사전 구축하기

실습 1_Word_Embedding.ipynb

- **데이터** : 한국어 위키백과 일부

> **위키백과**(Wiki百科, IPA: [ɰikɕibɛ̝̊kʰ̚kʰwa], [ykçibɛ̝̊kʰ̚kʰwa] (듣기)) 혹은 **위키피디아**(영어: Wikipedia 위키피디어[*], IPA: [ˌwɪkɪˈpiːdɪə] (듣기))는 누구나 자유롭게 쓸 수 있는 다언어판 인터넷 백과사전이다.[2] 2001년 1월 15일 지미 웨일스와 래리 생어가 시작하였으며[3], 대표적인 집단 지성의 사례로 평가받고 있다.[4]
>
> 위키백과는 자유 저작물을 보유하고 상업적인 광고가 없으며 주로 기부금을 통해 지원을 받는 비영리 단체인 위키미디어 재단에 의해 소유되고 지원을 받고 있다.[5][6][7][8] 2020년 기준으로 영어판 600만여 개, 한국어판 494,178개를 비롯하여 300여 언어판을 합하면 4천만 개 이상의 글이 수록되어 꾸준히 성장하고 있으며 앞으로 더 성장할 예정이다.[9] 위키백과의 저작권은 크리에이티브 커먼즈 라이선스(CCL)와 GNU 자유 문서(GFDL)의 2중 라이선스를 따른다. 두 라이선스 모두 자유 콘텐츠를 위한 것으로 일정한 요건을 갖추면 사용에 제약을 받지 않는다.

- **학습 목표** :
  - 자연어처리의 기본인 토크나이징을 이해한다.
  - 딥러닝을 위한 단어사전을 구축할 때 유의할 점을 파악한다.

실습 2. CBOW 알고리즘은 이용해 단어 임베딩은 만들어 봅시다!

- **데이터** : 한국어 위키백과 일부

위키백과(Wiki百科, IPA: [ɰikɕibɛ̝̈kʰ̬wa], [ykçibɛ̝̈kʰ̬wa] (듣기)) 혹은 **위키피디아**(영어: Wikipedia 위키피디어[*], IPA: [ˌwɪkɪˈpiːdɪə] (듣기))는 누구나 자유롭게 쓸 수 있는 다언어판 인터넷 백과사전이다.[2] 2001년 1월 15일 지미 웨일스와 래리 생어가 시작하였으며[3], 대표적인 집단 지성의 사례로 평가받고 있다.[4]

위키백과는 자유 저작물을 보유하고 상업적인 광고가 없으며 주로 기부금을 통해 지원을 받는 비영리 단체인 위키미디어 재단에 의해 소유되고 지원을 받고 있다.[5][6][7][8] 2020년 기준으로 영어판 600만여 개, 한국어판 494,178개를 비롯하여 300여 언어판을 합하면 4천만 개 이상의 글이 수록되어 꾸준히 성장하고 있으며 앞으로 더 성장할 예정이다.[9] 위키백과의 저작권은 크리에이티브 커먼즈 라이선스(CCL)와 GNU 자유 문서(GFDL)의 2중 라이선스를 따른다. 두 라이선스 모두 자유 콘텐츠를 위한 것으로 일정한 요건을 갖추면 사용에 제약을 받지 않는다.

- **학습 목표** :
  - CBOW 알고리즘과 그 특성을 이해한다.
  - 한국어 코퍼스로 CBOW 알고리즘을 학습하고 워드 벡터를 추출할 수 있다.