

Natural Language Processing with PyTorch

파이토치로 배우는 자연어 처리



딥러닝을 이용한
자연어 처리
애플리케이션 구축

1 ~ 2장 발표

24.07.29
김태균

목차

1. 소개

1.1 딥러닝 관련 용어

1.2 파이토치 기초

2. NLP 배경지식

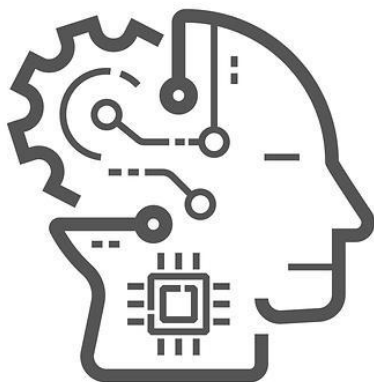
1. 소개

1.0 교재

목표

- 딥러닝(Deep Learning)

: 계산 그래프와 수치 최적화 기술을 사용해
데이터에서 표현을 효과적으로 학습하는 기술



- NLP(Natural Language Processing)

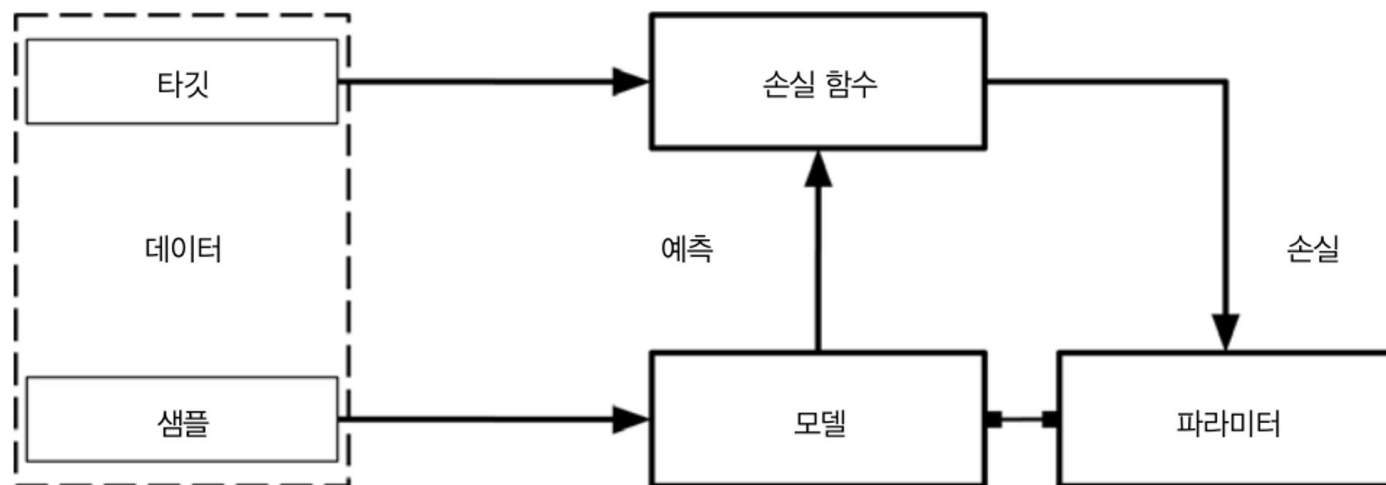
: 텍스트를 이해하는 통계적인 방법을 사용해
실전 문제를 해결하는 일련의 기술



1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

: 레이블된 훈련 샘플로 학습

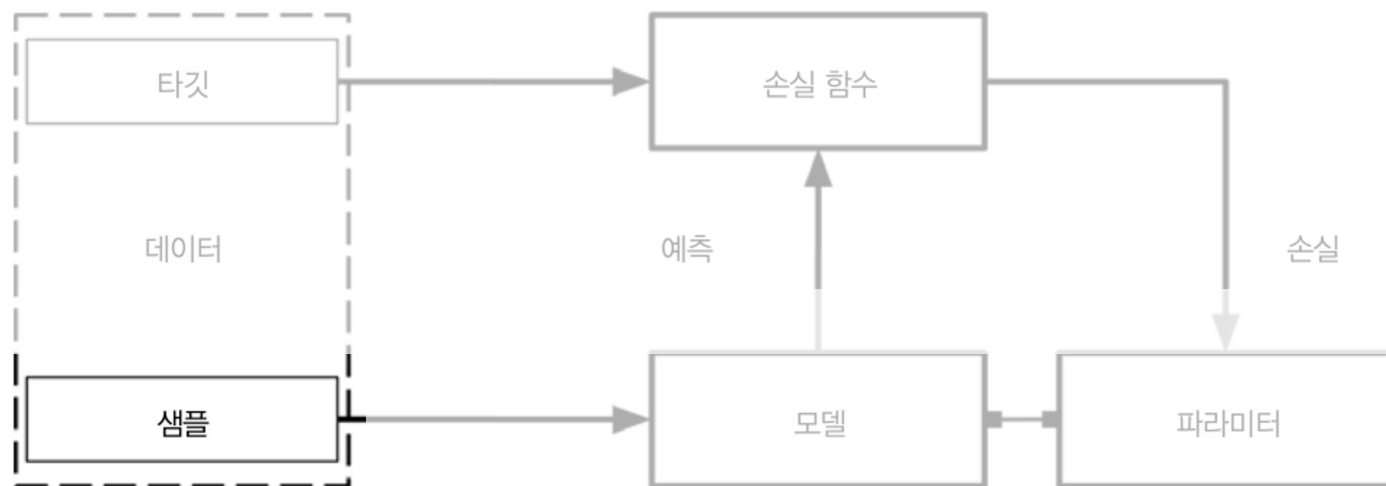


1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

: 레이블된 훈련 샘플로 학습

- 샘플 : 예측에 사용하는 아이템

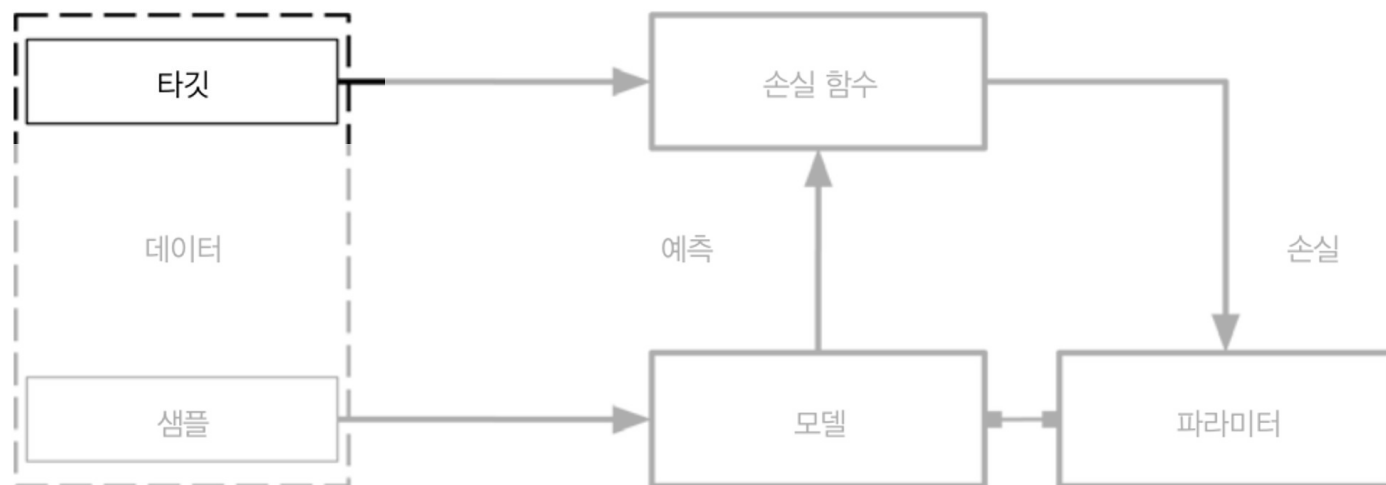


1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

: 레이블된 훈련 샘플로 학습

- 타깃 : 예측 대상

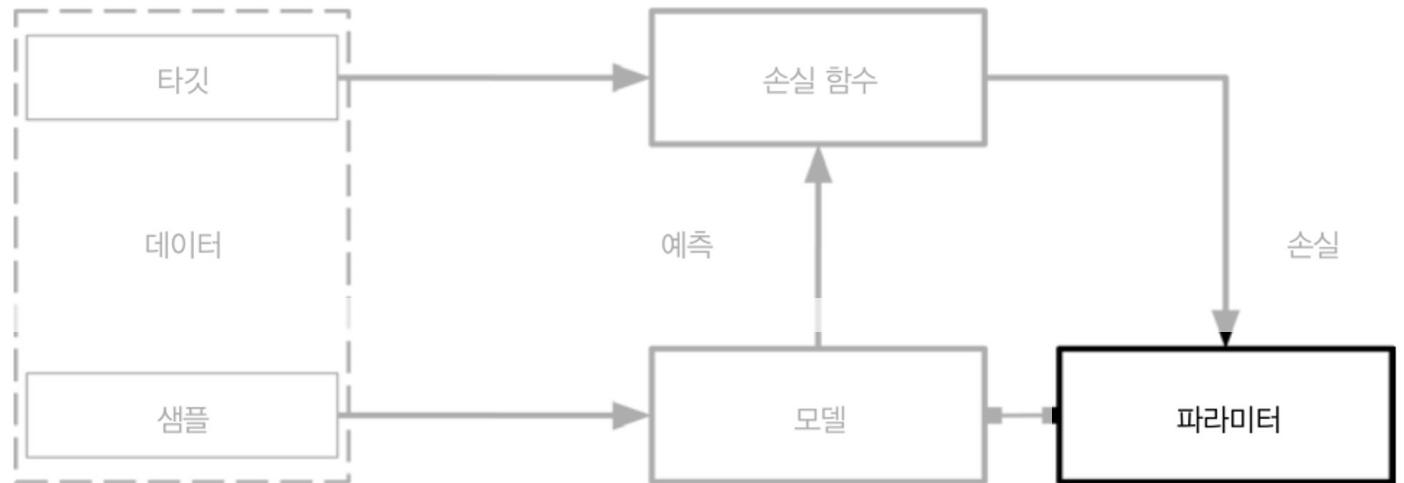


1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

: 레이블된 훈련 샘플로 학습

- 파라미터 : 모델을 규정하는 가중치

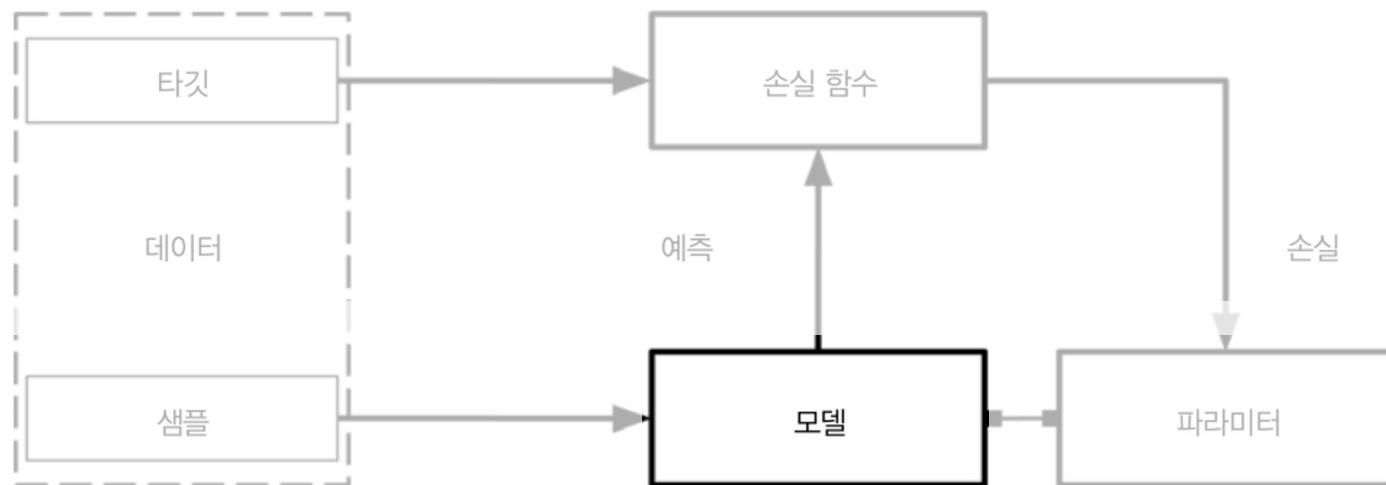


1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

: 레이블된 훈련 샘플로 학습

- 모델 : 타깃을 예측하는 함수

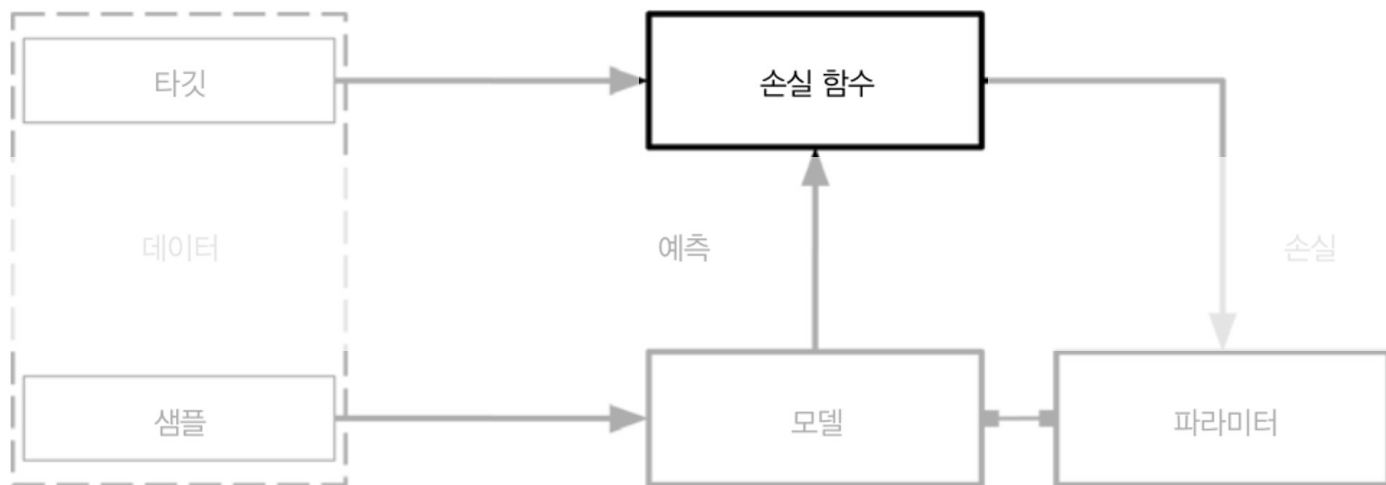


1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

: 레이블된 훈련 샘플로 학습

- 손실 함수 : 훈련 데이터에 대한 예측이 타겟과 얼마나 멀리 떨어져 있는지 비교하는 함수



1.1 딥러닝 관련 용어

- 지도 학습(Supervised Learning)

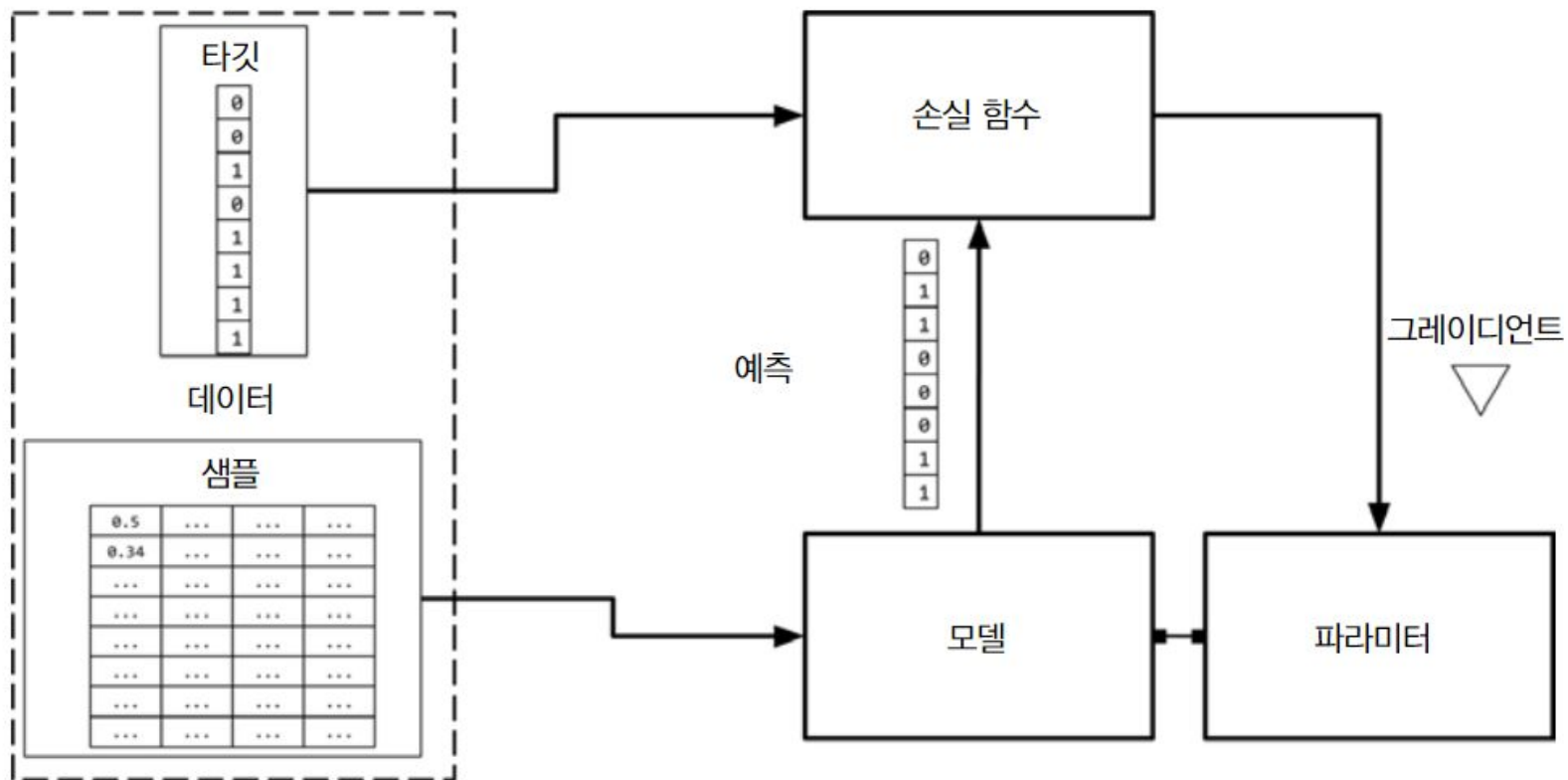
: 레이블된 훈련 샘플로 학습

- 목적 : 샘플 전체의 누적 손실을 최소화하는 **최적의 파라미터 찾기**
 - 경사 하강법 : 손실 함수의 기울기가 가장 작은 지점을 찾는 과정
 - 역전파 : 출력층에서 발생한 오류를 통해 각 층의 파라미터를 업데이트
 - 정방향 계산 : 현재 파라미터 값으로 입력을 평가하여 손실 함수를 계산
 - 역방향 계산 : 손실의 gradient를 사용하여 파라미터를 업데이트

1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

- 원-핫 표현
- TF-IDF 표현
- 타겟 인코딩



1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

- 1. 원-핫 표현

: 해당 범주에 해당하는 인덱스만 1로 설정하고

나머지 인덱스는 모두 0으로 설정

Time flies like an arrow.

Fruit flies like a banana.

	time	fruit	flies	like	a	an	arrow	banana
1 time	1	0	0	0	0	0	0	0
1 fruit	0	1	0	0	0	0	0	0
1 flies	0	0	1	0	0	0	0	0
1 like	0	0	0	1	0	0	0	0
1 a	0	0	0	0	1	0	0	0
1 an	0	0	0	0	0	1	0	0
1 arrow	0	0	0	0	0	0	1	0
1 banana	0	0	0	0	0	0	0	1

1.1 딥러닝 관련 용어

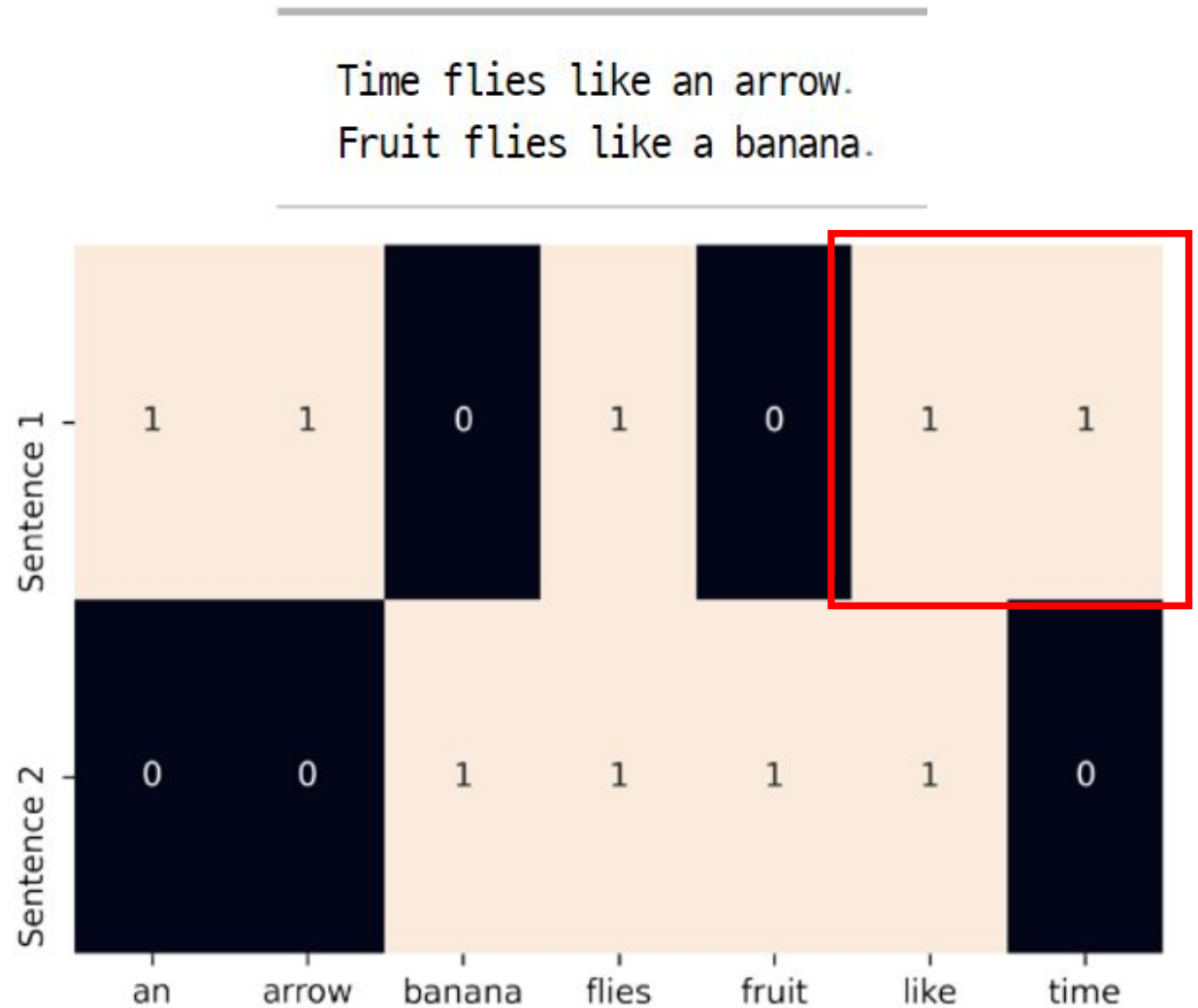
- 샘플과 타겟의 인코딩

2. TF-IDF 표현

– TF(Term Frequency)

: 문서 내에서 특정 단어가

얼마나 자주 등장하는지를 측정하는 방법



1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

2. TF-IDF 표현

– IDF(Inverse Document Frequency)

: 단어가 특정 문서에서

얼마나 드물게 등장하는지를 측정하는 방법

$$IDF(w) = \log \frac{N+1}{n_w+1} + 1$$

N : 전체 문서의 개수,

n_w : w 를 포함한 문서의 개수

1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

2. TF-IDF 표현

: 특정 단어가 문서 내에서 얼마나 중요한지를 측정하는 방법

$$TF(w) * IDF(w)$$

1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

2. TF-IDF 표현

: 특정 단어가 문서 내에서 얼마나 중요한지를 측정하는 방법

Time flies like an arrow.
Fruit flies like a banana.

$$TF(w) * IDF(w)$$

Ex) 첫 번째 문장의 'like'와 'time' 비교

$$TF('like') = 1, TF('time') = 1$$

$$IDF('like') = \log \frac{2+1}{2+1} + 1 = 1$$

$$IDF('time') = \log \frac{2+1}{1+1} + 1 = 1.41$$

1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

2. TF-IDF 표현

: 특정 단어가 문서 내에서 얼마나 중요한지를 측정하는 방법

Time flies like an arrow.
Fruit flies like a banana.

$$TF(w) * IDF(w)$$

Ex) 첫 번째 문장의 'like'와 'time' 비교

$$TF-IDF('like') = 1 * 1 = 1$$

$$TF-IDF('time') = 1 * 1.41 = 1.41$$

1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

2. TF-IDF 표현

: 특정 단어가 문서 내에서 얼마나 중요한지를 측정하는 방법

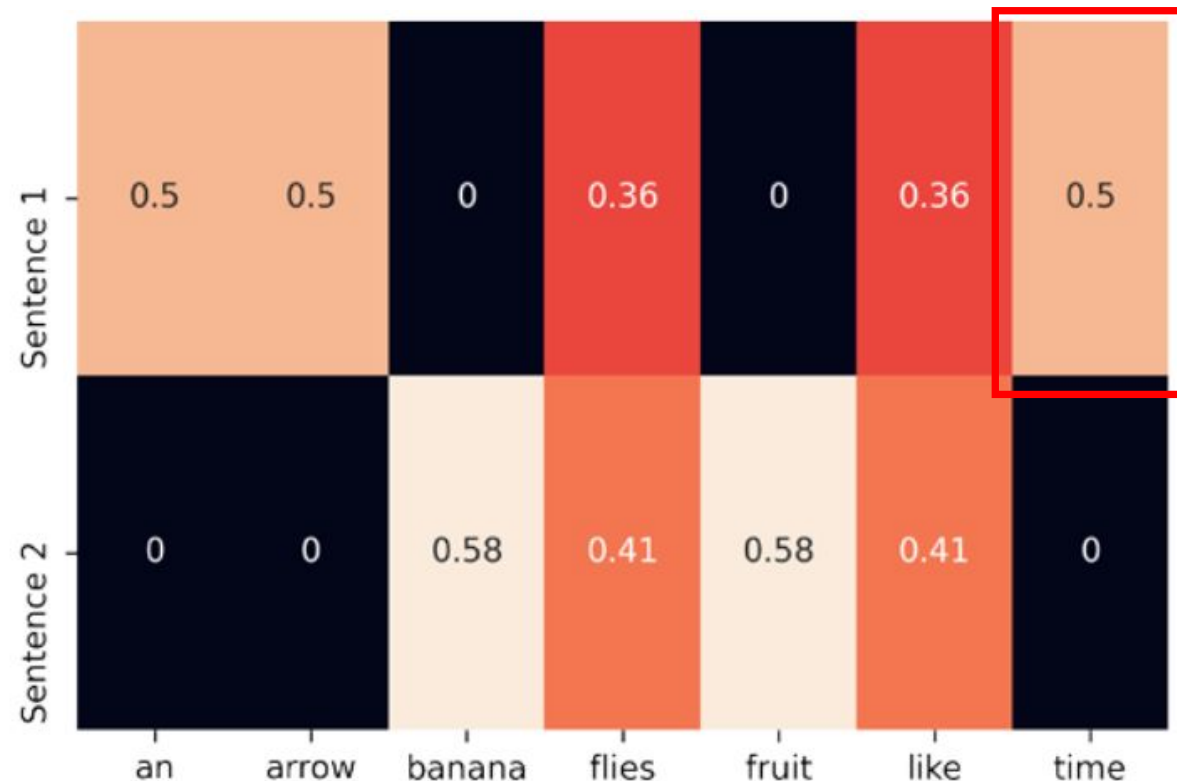
$$TF(w) * IDF(w)$$

Ex) 첫 번째 문장의 'like'와 'time' 비교

- L2 norm 정규화

'like' \doteq 0.36, 'time' \doteq 0.5

Time flies like an arrow.
Fruit flies like a banana.



1.1 딥러닝 관련 용어

- 샘플과 타겟의 인코딩

3. 타겟 인코딩

: 범주형 변수가 타겟과 어떻게 관련되어 있는지를 반영하기 위해
수치형 변수로 변환하는 기법

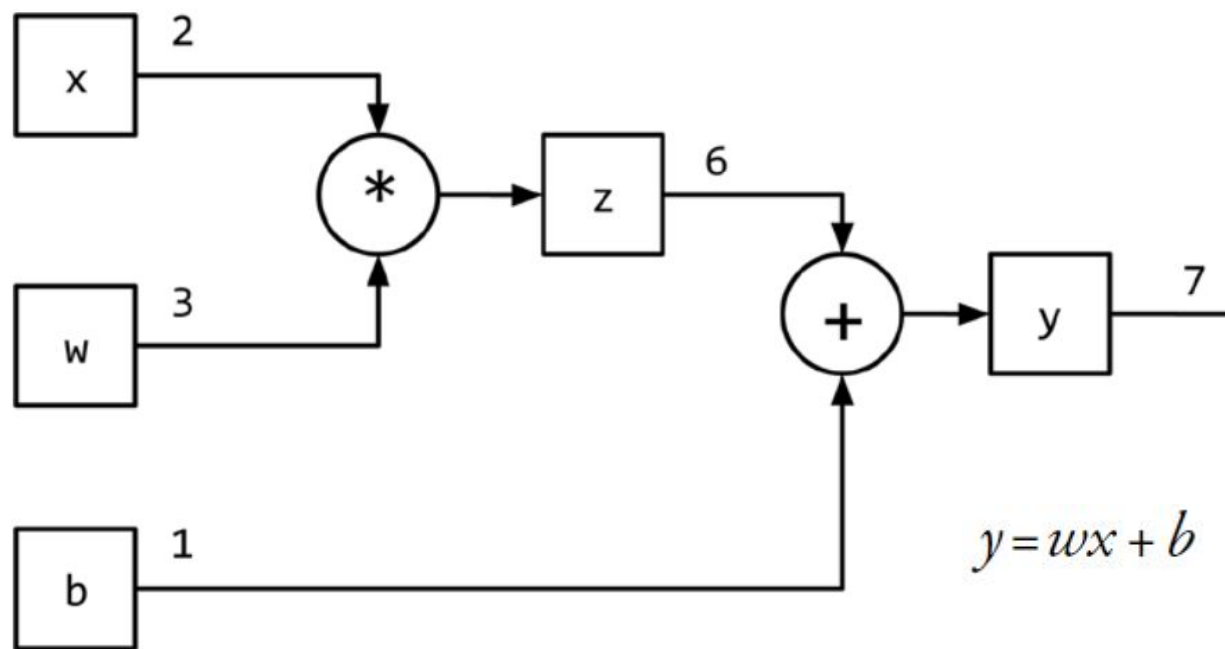
1.1 딥러닝 관련 용어

- 계산 그래프

: 수학적 연산과 그 연산의 종속성을 나타내는 그래프 구조

- 구성 요소

- 노드 : 수학 연산을 나타냄
- 엣지 : 데이터의 흐름을 나타냄



1.2 파이토치 기초

- Pytorch

: 오픈 소스 딥러닝 프레임워크

– 동적 계산 그래프와 정적 계산 그래프

특징	동적 계산 그래프(Pytorch)	정적 계산 그래프(TensorFlow)
그래프 생성 시점	실행 시 동적으로 생성	미리 정의된 후 실행
유연성	높은 유연성	그래프 고정

1.2 파이토치 기초

- Pytorch

: 오픈 소스 딥러닝 프레임워크

– 동적 계산 그래프와 정적 계산 그래프

특징	동적 계산 그래프(Pytorch)	정적 계산 그래프(TensorFlow)
그래프 생성 시점	실행 시 동적으로 생성	미리 정의된 후 실행
유연성	높은 유연성	그래프 고정

1.2 파이토치 기초

- Pytorch

: 오픈 소스 딥러닝 프레임워크

- 텐서 : 다차원 데이터를 담은 수학 객체

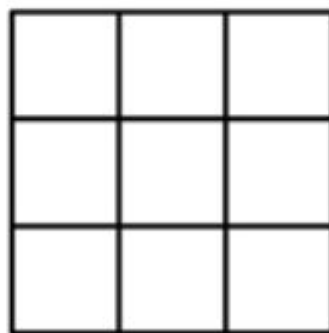
스칼라
랭크 0 텐서



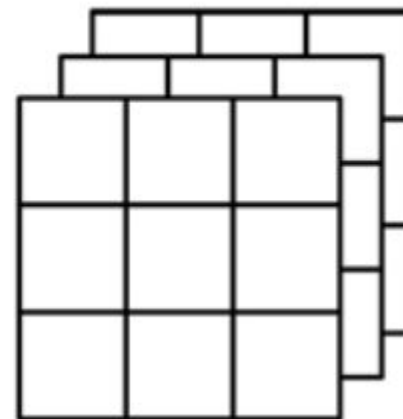
벡터
랭크 1 텐서



행렬
랭크 2 텐서



랭크 3 텐서



1.2 파이토치 기초

- 예제

1. 텐서 생성 및 초기화
2. 인덱싱, 슬라이싱,
연결
3. 행렬 곱셈
4. 그레디언트 연산

1.2 파이토치 기초

- 예제

1. 텐서 생성 및 초기화

Ex) 랜덤으로 초기화된 2차원 텐서

```
1  import torch
2
3
4  x = torch.rand(2, 3)
5  y = torch.zeros(2, 3)
6  z = torch.ones(2, 3)
7  print(x, y, z, sep='\n')
```

```
tensor([[0.1733, 0.4065, 0.0719],
        [0.1351, 0.3164, 0.3728]])
tensor([[0., 0., 0.],
        [0., 0., 0.]])
tensor([[1., 1., 1.],
        [1., 1., 1.]])
```

1.2 파이토치 기초

- 예제

2. 인덱싱, 슬라이싱, 연결

Ex) 0~5의 값을 갖는 2차원

텐서

```
4 x = torch.arange(6).view(2, 3)
5 print(x)
6 print(x[:1, :2])
7 print(x[0, 1])
```

```
tensor([[0, 1, 2],
        [3, 4, 5]])
tensor([[0, 1]])
tensor(1)
```

1.2 파이토치 기초

- 예제

2. 인덱싱, 슬라이싱, 연결

Ex) 0~5의 값을 갖는 2차원

텐서

```
4 x = torch.arange(6).view(2, 3)
5 indices = torch.LongTensor([0, 2])
6 print(x)
7 print(torch.index_select(x, dim=1, index=indices))
```

```
tensor([[0, 1, 2],
        [3, 4, 5]])
tensor([[0, 2],
        [3, 5]])
```

1.2 파이토치 기초

- 예제

2. 인덱싱, 슬라이싱, 연결

Ex) 0~5의 값을 갖는 2차원

텐서

```
4 x = torch.arange(6).view(2, 3)
5 print(x)
6 print(torch.cat([x, x], dim=0))
7 print(torch.cat([x, x], dim=1))
8 print(torch.stack([x, x]))
```

```
tensor([[0, 1, 2],
        [3, 4, 5]])
tensor([[0, 1, 2],
        [3, 4, 5],
        [0, 1, 2],
        [3, 4, 5]])
tensor([[0, 1, 2, 0, 1, 2],
        [3, 4, 5, 3, 4, 5]])
tensor([[[0, 1, 2],
         [3, 4, 5]],

        [[0, 1, 2],
         [3, 4, 5]]])
```

1.2 파이토치 기초

- 예제

3. 행렬 곱셈

Ex) 0~5의 값을 갖는 2차원

텐서

```
4 x = torch.arange(6).view(2, 3).float()
5 y = torch.ones(3, 2)
6 y[:, 1] += 1
7 print(x)
8 print(y)
9 print(torch.mm(x, y))
```

```
tensor([[0., 1., 2.],
        [3., 4., 5.]])
tensor([[1., 2.],
        [1., 2.],
        [1., 2.]])
tensor([[ 3.,  6.],
        [12., 24.]])
```

1.2 파이토치 기초

- 예제

4. 그래디언트 연산

Ex) 2차원 텐서의 그래디언트

```
4 x = torch.ones(2, 2, requires_grad=True)
5 print(f'역방향 계산 전:\n {x.grad}')
6
7 y = (x + 2) * (x + 5) + 3
8 z = y.mean()
9 z.backward()
10 print(f'역방향 계산 후:\n {x.grad}')
```

```
역방향 계산 전:
None
역방향 계산 후:
tensor([[2.2500, 2.2500],
        [2.2500, 2.2500]])
```

2. NLP 배경지식

배경지식

: 컴퓨터와 언어의 상호작용을 연구하는 학문 분야로, 언어의 특징을 이해하는 방법을 개발

cf) NLP : 실용적인 문제를 해결하는 방법을 개발



2. NLP

배경지식

- 말뭉치(corpus)

: 텍스트 데이터

- 토큰(token)

: 텍스트 데이터를 처리하기 위해

작은 단위로 분할한 조각

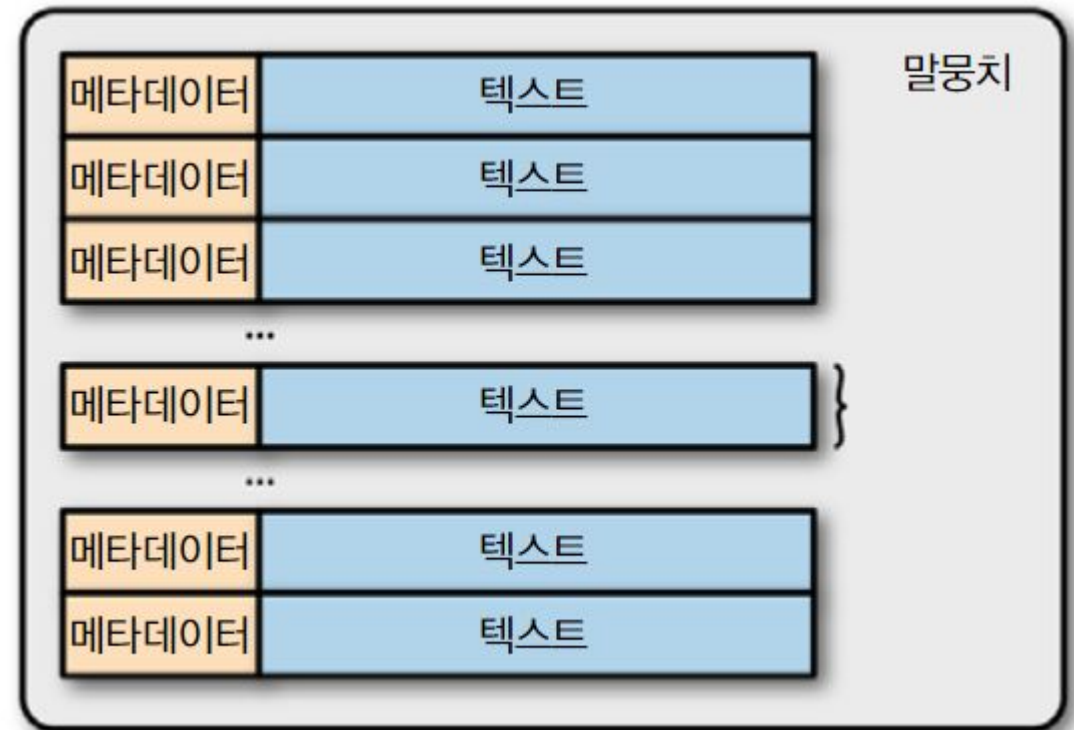
- 타입(type)

: 텍스트 데이터의 고유한 형태

- 특성 공학(feature engineering)

: 원본 데이터를 모델 학습에

유용하게 사용될 수 있는 형태로 변환하는 과정



2. NLP

배경지식

- 텍스트

투크하

```
In[0] import spacy
      nlp = spacy.load('en')
      text = "Mary, don't slap the green witch"
      print([str(token) for token in nlp(text.lower())])
```

```
Out[0] ['mary', ',', 'do', "n't", 'slap', 'the', 'green', 'witch', '.']
```

```
In[1] from nltk.tokenize import TweetTokenizer
      tweet=u"Snow White and the Seven Degrees
            #MakeAMovieCold@midnight:-)"
      tokenizer = TweetTokenizer()
      print(tokenizer.tokenize(tweet.lower()))
```

```
Out[1] ['snow', 'white', 'and', 'the', 'seven', 'degrees',
        '#makeamoviecold', '@midnight', ':~)']
```

2. NLP


배경지식

- N-gram

: 텍스트에 있는 n 개의 고정 길이의 연속된 토큰 시퀀스

```
In[0] def n_grams(text, n):  
    ...  
    takes tokens or text, returns a list of n-grams  
    ...  
    return [text[i:i+n] for i in range(len(text)-n+1)]  
  
cleaned = ['mary', ',', 'n't', 'slap', 'green', 'witch', '.']  
print(n_grams(cleaned, 3))
```

```
Out[0] [['mary', ',', 'n't'],  
        [',', 'n't', 'slap'],  
        ['n't', 'slap', 'green'],  
        ['slap', 'green', 'witch'],  
        ['green', 'witch', '.']]
```



2. NLP

배경지식

- 표제어 추출(lemmatization)


: 단어의 기본형을 추출

- 어간 추출(stemming)

: 접사를 제거한 단어의 어간을 추출

```
In[0] import spacy
      nlp = spacy.load('en')
      doc = nlp(u"he was running late")
      for token in doc:
          print('{} --> {}'.format(token, token.lemma_))
```

Out[0] he --> he
 was --> be
 running --> run
 late --> late



2. NLP

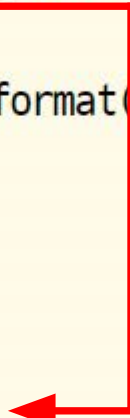
배경지식

- 품사 태깅(part-of-speech tagging)

: 문장 내의 각 단어에 대해 품사를 할당

```
In[0] import spacy
      nlp = spacy.load('en')
      doc = nlp(u"Mary slapped the green witch.")
      for token in doc:
          print('{} - {}'.format(token, token.pos_))
```

Out[0] Mary - PROPN
 slapped - VERB
 the - DET
 green - ADJ
 witch - NOUN
 . - PUNCT



2. NLP

배경지식

- 청크 나누기(chunking)

: 문장 내에서 주요 구문 단위로 추출

- 개체명 인식(named entity recognition)

: 명명된 개체를 인식하고 분류

```
In[0] import spacy
      nlp = spacy.load('en')
      doc = nlp(u"Mary slapped the green witch.")
      for chunk in doc.noun_chunks:
          print ('{} - {}'.format(chunk, chunk.label_))

Out[0] Mary - NP
      the green witch - NP
```

John PERSON was born in Chicken GPE , Alaska GPE , and studies at Cranberry Lemon University ORG .

2. NLP

배경지식

- 구문 분석(parsing)

: 구 사이의 관계를 파악하는 작업

- 구성 구문 분석(constituency parsing)
- 의존 구문 분석(dependency parsing)

2. NLP

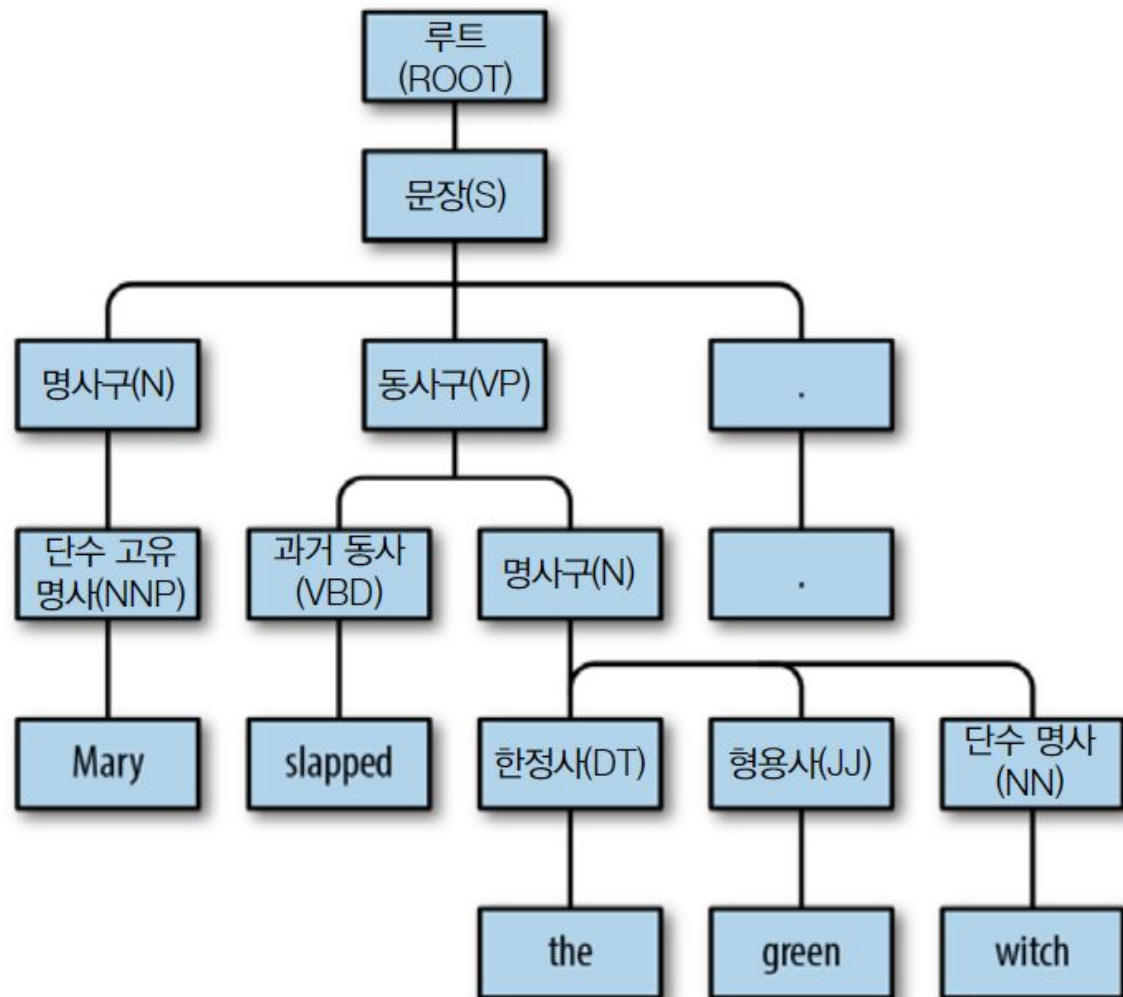
배경지식

- 구문 분석(parsing)

: 구 사이의 관계를 파악하는 작업

- 구성 구문 분석(constituency parsing)

: 구와 구의 계층적 관계로 분석



2. NLP

배경지식

- 구문 분석(parsing)

: 구 사이의 관계를 파악하는 작업

- 의존 구문 분석(dependency parsing)

: 단어 간의 관계 분석



정리

- 딥러닝 관련 용어
- 파이토치 기초
- NLP 배경지식

QnA