텍스트 데이터 다루기

학습 목표

- 텍스트를 처리하기 위한 원핫인코딩을 이해해 봅니다.

학습 내용

- 원핫 인코딩을 이해하고 실습해 봅니다.
- 케라스를 활용하여 원핫인코딩을 실습해 봅니다.
- 해싱 기법을 활용한 원핫 인코딩 실습을 해 봅니다.

목차

라이브러리 준비

In [1]:

```
import keras
import tensorflow as tf
print(keras.__version__)
print(tf.__version__)
```

2.9.0
2.9.1

01. 텍스트 전처리 이해

목차로 이동하기

텍스트 데이터

- 텍스트는 가장 흔한 시퀀스(순서가 있는) 형태의 데이터이다.
- 시퀀스 처리용 딥러닝 모델은 문서 분류, 감성 분석, 저자 식별, 질문 응답 등의 애플리케이션에 적합하다.
- 텍스트를 수치형 텐서로 변환하는 과정을 **텍스트 벡터화(vectorizing text)**라고 한다.
  - 방법1 : 텍스트를 **단어로 나누고** 각 단어를 하나의 벡터로 변환
  - 방법2 : 텍스트를 **문자로 나누고** 각 문자를 하나의 벡터로 변환
  - 방법3 : 텍스트에서 **단어나 문자의 n-그램(n-gram)추출**하여 각 n-그램을 하나의 벡터로 변환
    - n-gram은 n개의 연속적인 단어 나열을 의미
    - n-gram은 연속된 단어나 문자의 그룹이다. 텍스트에서 단어나 문자를 하나씩 이동하면서 추출
- 텍스트를 나누는 이런 단위(단어, 문자, n-그램)를 **토큰(token)**이라 한다.

    - 토큰으로 나누는 작업을 토큰화(tokenization)이라 한다.

텍스트의 벡터화 과정

- STEP 01 어떤 방법ㅇ로 토큰화를 진행할지 결정.
- STEP 02 토큰에 수치형 벡터를 연결
- STEP 03 벡터는 시퀀스 텐서로 묶여져서 **심층 신경망의 입력**으로 사용

토큰과 벡터를 연결하는 방법 2가지

- 원-핫 인코딩(one-hot encoding)
- 토큰 임베딩(token embedding) - 보통 단어에만 사용되면 워드임베딩(word embedding)이라 말한다.

02. 원핫 인코딩 실습해 보기

목차로 이동하기

원핫 인코딩 실습해 보기

- 모든 데이터에 대해서(문장)을 토큰으로 나눈다.
    - 나누어진 토큰을 이를 사전에 하나의 **단어와 인덱스**로 만든다.
- 단어 사전이 갖는 인덱스 수만큼의 배열을 만든다.(값은 0으로 채운다.) np.zeros이용.
- 단어들을 인덱스 값(0,10)을 원핫 인코딩하여 해당 위치에 1을 채운다.

단어사전 구축

In [3]:

```
import numpy as np

# 하나의 원소가 샘플. 하나의 문장.
samples = ['The cat sat on the mat.', 'The dog ate my homework.']

# 문장에 있는 단어들을 가지고 인덱스(사전)을 구축함.
# 단어:인덱스
token_index = {}
for sample in samples:
    for word in sample.split():
        if word not in token_index:
            token_index[word] = len(token_index) + 1

max_length = 10
token_index
```

Out[3]:

```
{'The': 1,
 'cat': 2,
 'sat': 3,
 'on': 4,
 'the': 5,
 'mat.': 6,
 'dog': 7,
 'ate': 8,
 'my': 9,
 'homework.': 10}
```

In [4]:

```
# 단어 인덱스의 길이 확인
max(token_index.values()) + 1
```

Out[4]:

```
11
```

단어의 수만큼 0의 값을 갖는 인덱스 배열 생성

In [5]:

```
# 결과를 저장할 배열(0으로 이루어진 벡터)
# max_length는 사용할 단어수
results = np.zeros((len(samples), max_length, max(token_index.values()) + 1))
print(results)
```

```
[[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

 [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]]
```

In [11]:

```
token_index.get(word)
```

Out[11]:

```
10
```

In [12]:

```
# 해당되는 단어 위치에 1로 치환
for i, sample in enumerate(samples):
    for j, word in list(enumerate(sample.split()))[:max_length]:
        index = token_index.get(word)
        results[i, j, index] = 1.  # 행, 열, 인덱스를 1로 치환
```

In [13]:

```
print(samples)
print(token_index)
print(results)
```

```
['The cat sat on the mat.', 'The dog ate my homework.']
{'The': 1, 'cat': 2, 'sat': 3, 'on': 4, 'the': 5, 'mat.': 6, 'dog': 7, 'ate': 8, 'm
y': 9, 'homework.': 10}
[[[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

 [[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]]
```

케라스(텐서플로워)를 활용한 원핫 인코딩 실습

- 케라스에서는 원본 텍스트 데이터를 단어 또는 문자 수준의 원-핫 인코딩으로 변환해 주는 유틸리티가 있다.

In [14]:

```
from keras.preprocessing.text import Tokenizer

samples = ['The cat sat on the mat.', 'The dog ate my homework.']
```

STEP 01. 단어들에 대한 인덱스 값 얻기

In [16]:

```
samples
```

Out[16]:

```
['The cat sat on the mat.', 'The dog ate my homework.']
```

- Tokenizer 클래스
  - 문장으로부터 단어를 토큰화하고 숫자에 대응시키는 딕셔너리를 사용할 수 있도록 함.
  - fit_on_texts() : 문자 데이터를 입력받아, 리스트 형태로 변환
  - texts_to_sequences() : 메서드를 이용하여 단어들을 시퀀스 형태로 변환.

In [18]:

```
# 가장 빈도가 높은 1,000개의 단어만 선택하도록 Tokenizer 객체를 만듭니다.
tokenizer = Tokenizer(num_words=1000)

# 단어 인덱스를 구축합니다.
tokenizer.fit_on_texts(samples)
word_idx = tokenizer.word_index
print("단어 인덱스 : ", word_idx)

# 문자열을 정수 인덱스의 리스트로 변환합니다.
sequences = tokenizer.texts_to_sequences(samples)
sequences
```

```
단어 인덱스 :  {'the': 1, 'cat': 2, 'sat': 3, 'on': 4, 'mat': 5, 'dog': 6, 'ate': 7,
'my': 8, 'homework': 9}
```

Out[18]:

```
[[1, 2, 3, 4, 1, 5], [1, 6, 7, 8, 9]]
```

STEP 02. 단어들의 시퀀스를 원핫 이진 벡터 표현

- 각각의 문장을 1000여개의 단어의 위치에 1로 매핑시키기.

In [19]:

```
one_hot = tokenizer.texts_to_matrix(samples, mode='binary')

print(one_hot.shape)
print(one_hot)
```

```
(2, 1000)
[[0. 1. 1. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]]
```

In [12]:

```
# 계산된 단어 인덱스를 구합니다.
word_index = tokenizer.word_index
print(word_index)
print('Found %s unique tokens.' % len(word_index))
```

```
{'the': 1, 'cat': 2, 'sat': 3, 'on': 4, 'mat': 5, 'dog': 6, 'ate': 7, 'my': 8, 'home
work': 9}
Found 9 unique tokens.
```

03 해싱 기법을 이용한 원핫 인코딩-변형된 형태

목차로 이동하기

- 원핫 인코딩의 변종중의 하나는 원-핫 해싱(one-hot hashing)기법입니다.
  - 해싱 기법
    - 동일한 인풋은 동일한 결과(숫자)가 나온다.

    - 결과를 바탕으로 인풋을 다시 찾을 수 없음.
    - 다른 인풋을 넣어도 같은 결과가 나올 수 있다. 이를 Hash Collision이라 한다.
- 고유한 토큰의 수가 너무 커서 다루기 어려울 때 사용.
    - 각 단어에 명시적으로 인덱스를 할당하고 이 인덱스를 딕셔너리에 저장하지 않고, 단어를 해싱한다.

In [20]:

```
samples = ['The cat sat on the mat.', 'The dog ate my homework.']

# 단어를 크기가 1,000인 벡터로 저장합니다.
# 1,000개(또는 그이상)의 단어가 있다면 해싱 충돌이 늘어나고 인코딩의 정확도가 감소될 것입니다
dimensionality = 1000
max_length = 10
```

In [23]:

```
results = np.zeros((len(samples), max_length, dimensionality))
print(results.shape)

for i, sample in enumerate(samples):
    print(sample)
    for j, word in list(enumerate(sample.split()))[:max_length]:
        # 단어를 해싱하여 0과 1,000 사이의 랜덤한 정수 인덱스로 변환합니다.
        index = abs(hash(word)) % dimensionality
        results[i, j, index] = 1.
```

```
(2, 10, 1000)
The cat sat on the mat.
The dog ate my homework.
```

In [22]:

```
results.shape, results
```

Out[22]:

```
((2, 10, 1000),
 array([[[0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         ...,
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.]],

        [[0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         ...,
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.],
         [0., 0., 0., ..., 0., 0., 0.]]]))
```