텍스트 데이터 다루기

In [1]:
```
import keras
keras.__version__
```

Out[1]: '2.4.3'

원핫 인코딩 실습해 보기

- 모든 데이터에 대해서(문장)을 토큰으로 나눈다. 그리고 이를 사전에 하나의 단어와 인덱스로 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.

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

In [6]:
```
# 단어 인덱스의 길이 확인
max(token_index.values()) + 1
```

Out[6]: 11

In [7]:
```
# 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.]]]
```

In [11]:
```
token_index.get(word)
```

Out[11]: 10

In [8]:
```
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 [9]:
```
results
```

Out[9]:
```
array([[[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.],
        [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]]])
```

케라스를 활용한 원핫 인코딩 실습

In [12]:
```
from keras.preprocessing.text import Tokenizer

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

2-1 단어들에 대한 인덱스 값 얻기

In [13]:
```
# 가장 빈도가 높은 1,000개의 단어만 선택하도록 Tokenizer 객체를 만듭니다.
tokenizer = Tokenizer(num_words=1000)
```

```
# 단어 인덱스를 구축합니다.
tokenizer.fit_on_texts(samples)

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

Out[13]: [[1, 2, 3, 4, 1, 5], [1, 6, 7, 8, 9]]

2-2 직접 원핫 이진 벡터 표현 얻기

- 인덱스된 값을 1000여개의 단어의 해당 단어의 위치에 매핑

In [16]:
```
one_hot = tokenizer.texts_to_matrix(samples, mode='binary')
one_hot.shape, one_hot
```

Out[16]:
```
((2, 1000),
 array([[0., 1., 1., ..., 0., 0., 0.],
        [0., 1., 0., ..., 0., 0., 0.]]))
```

In [18]:
```
# 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, 'homewo
rk': 9}
Found 9 unique tokens.
```

03 해싱 기법을 이용한 원핫 인코딩-변형된 형탱

In [19]:
```
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 
dimensionality = 1000
max_length = 10

results = np.zeros((len(samples), max_length, dimensionality))
for i, sample in enumerate(samples):
    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.
```

In [21]:
```
results.shape, results
```

Out[21]:
```
((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.]]]))
```

In [ ]: