로이터 데이터셋 로드하기

In [1]:

```
from keras.datasets import reuters

(train_data, train_labels), (test_data, test_labels) = reuters.load_data(num_words=10000)
```

```
Using TensorFlow backend.
```

In [2]:

```
print(len(train_data))
print(len(test_data))
```

```
8982
2246
```

In [3]:

```
train_data[10]
```

Out[3]:

```
[1,
 245,
 273,
 207,
 156,
 53,
 74,
 160,
 26,
 14,
 46,
 296,
 26,
 39,
 74,
 2979,
 3554,
 14,
 46,
 4689,
 4329,
 86,
 61,
 3499,
 4795,
 14,
 61,
 451,
 4329,
 17,
 12]
```

로이터 데이터셋을 텍스트로 디코딩하기

In [4]:

```
word_index = reuters.get_word_index()
reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
decoded_newswire = ' '.join([reverse_word_index.get(i - 3, '?') for i in train_data[0]])
```

In [5]:

```
train_labels[10]
```

Out[5]:

```
3
```

데이터 인코딩하기

In [6]:

```
import numpy as np

def vectorize_sequences(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results

x_train = vectorize_sequences(train_data)
x_test = vectorize_sequences(test_data)
```

In [24]:

```
def to_one_hot(labels, dimension=46):
    results = np.zeros((len(labels), dimension))
    for i, labels in enumerate(labels):
        results[i, labels] = 1
    return results

one_hot_train_labels = to_one_hot(train_labels)
one_hot_test_labels = to_one_hot(test_labels)
```

In [8]:

```
from keras.utils.np_utils import to_categorical

one_hot_train_labels = to_categorical(train_labels)
one_hot_test_labels = to_categorical(test_labels)
```

In [25]:

```
from keras import models, layers

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='softmax'))
```

In [10]:

```
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [11]:

```
x_val = x_train[:1000]
partial_x_train = x_train[1000:]
y_val = one_hot_train_labels[:1000]
partial_y_train = one_hot_train_labels[1000:]
```

In [12]:

```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data = (x_val, y_val))
```

```
WARNING:tensorflow:From C:₩Users₩J₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩o
ps₩math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_ops) is deprecated
and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 7982 samples, validate on 1000 samples
Epoch 1/20
7982/7982 [==============================] - 3s 315us/step - loss: 2.5322 - acc:
0.4955 - val_loss: 1.7208 - val_acc: 0.6120
Epoch 2/20
7982/7982 [==============================] - 1s 84us/step - loss: 1.4452 - acc: 0.
6879 - val_loss: 1.3459 - val_acc: 0.7060
Epoch 3/20
7982/7982 [==============================] - 1s 84us/step - loss: 1.0953 - acc: 0.
7651 - val_loss: 1.1708 - val_acc: 0.7430
Epoch 4/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.8697 - acc: 0.
8165 - val_loss: 1.0793 - val_acc: 0.7590
Epoch 5/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.7034 - acc: 0.
8472 - val_loss: 0.9844 - val_acc: 0.7810
Epoch 6/20
7982/7982 [==============================] - 1s 85us/step - loss: 0.5667 - acc: 0.
8802 - val_loss: 0.9411 - val_acc: 0.8040
Epoch 7/20
7982/7982 [==============================] - 1s 86us/step - loss: 0.4581 - acc: 0.
9048 - val_loss: 0.9083 - val_acc: 0.8020
Epoch 8/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.3695 - acc: 0.
9231 - val_loss: 0.9363 - val_acc: 0.7890
Epoch 9/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.3032 - acc: 0.
9315 - val_loss: 0.8917 - val_acc: 0.8090
Epoch 10/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.2537 - acc: 0.
9414 - val_loss: 0.9071 - val_acc: 0.8110
Epoch 11/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.2187 - acc: 0.
9471 - val_loss: 0.9177 - val_acc: 0.8130
Epoch 12/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.1873 - acc: 0.
9508 - val_loss: 0.9027 - val_acc: 0.8130
Epoch 13/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.1703 - acc: 0.
9521 - val_loss: 0.9323 - val_acc: 0.8110
Epoch 14/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.1536 - acc: 0.
9554 - val_loss: 0.9689 - val_acc: 0.8050
Epoch 15/20
7982/7982 [==============================] - 1s 85us/step - loss: 0.1390 - acc: 0.
9560 - val_loss: 0.9686 - val_acc: 0.8150
Epoch 16/20
7982/7982 [==============================] - 1s 85us/step - loss: 0.1313 - acc: 0.
9560 - val_loss: 1.0220 - val_acc: 0.8060
Epoch 17/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.1217 - acc: 0.
9579 - val_loss: 1.0254 - val_acc: 0.7970
Epoch 18/20
7982/7982 [==============================] - 1s 85us/step - loss: 0.1198 - acc: 0.
9582 - val_loss: 1.0430 - val_acc: 0.8060
Epoch 19/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.1138 - acc: 0.
9597 - val_loss: 1.0955 - val_acc: 0.7970
Epoch 20/20
7982/7982 [==============================] - 1s 84us/step - loss: 0.1111 - acc: 0.
9593 - val_loss: 1.0674 - val_acc: 0.8020
```

In [14]:

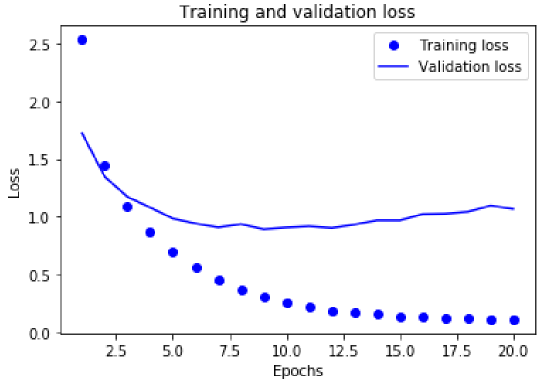
```python
import matplotlib.pyplot as plt

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

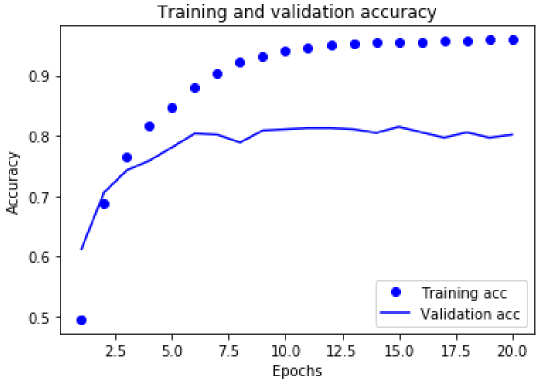


In [15]:

```
plt.clf()
acc = history.history['acc']
val_acc = history.history['val_acc']

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [16]:

```
from keras import models, layers

model = models.Sequential()
model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(46, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(partial_x_train,
                  partial_y_train,
                  epochs=9,
                  batch_size=512,
                  validation_data = (x_val, y_val))
results = model.evaluate(x_test, one_hot_test_labels)
```

```
Train on 7982 samples, validate on 1000 samples
Epoch 1/9
7982/7982 [==============================] - 1s 115us/step - loss: 3.3043 - acc:
0.4400 - val_loss: 2.8549 - val_acc: 0.5160
Epoch 2/9
7982/7982 [==============================] - 1s 85us/step - loss: 2.4160 - acc: 0.
4456 - val_loss: 2.0822 - val_acc: 0.3550
Epoch 3/9
7982/7982 [==============================] - 1s 85us/step - loss: 1.7412 - acc: 0.
3529 - val_loss: 1.6212 - val_acc: 0.3570
Epoch 4/9
7982/7982 [==============================] - 1s 84us/step - loss: 1.3375 - acc: 0.
4053 - val_loss: 1.3278 - val_acc: 0.6640
Epoch 5/9
7982/7982 [==============================] - 1s 84us/step - loss: 0.8727 - acc: 0.
8006 - val_loss: 1.0503 - val_acc: 0.7780
Epoch 6/9
7982/7982 [==============================] - 1s 85us/step - loss: 0.6391 - acc: 0.
8667 - val_loss: 0.9283 - val_acc: 0.8140
Epoch 7/9
7982/7982 [==============================] - 1s 85us/step - loss: 0.5074 - acc: 0.
8985 - val_loss: 0.9482 - val_acc: 0.8230
Epoch 8/9
7982/7982 [==============================] - 1s 86us/step - loss: 0.4239 - acc: 0.
9149 - val_loss: 0.9127 - val_acc: 0.8220
Epoch 9/9
7982/7982 [==============================] - 1s 86us/step - loss: 0.3549 - acc: 0.
9243 - val_loss: 0.9706 - val_acc: 0.8090
2246/2246 [==============================] - 0s 119us/step
```

In [17]:

```
results
```

Out[17]:

```
[1.0321700950870532, 0.7845057881207521]
```

In [22]:

```
import copy

test_labels_copy = copy.copy(test_labels)
np.random.shuffle(test_labels_copy)
hits_array = np.array(test_labels) == np.array(test_labels_copy)
float(np.sum(hits_array)) / len(test_labels)
```

Out[22]:

```
0.19056099732858414
```

In [18]:

```
predictions = model.predict(x_test)

# predictions[0].shape
# np.sum(predictions[0])
# np.argmax(predictions[0])
```

In [19]:

```
np.sum(predictions[0])
```

Out[19]:

```
0.0005772114
```

In [20]:

```
from keras import models, layers

model = models.Sequential()
model.add(layers.Dense(128, activation='relu', input_shape=(10000,)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(46, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(partial_x_train,
                  partial_y_train,
                  epochs=9,
                  batch_size=512,
                  validation_data = (x_val, y_val))
results = model.evaluate(x_test, one_hot_test_labels)
```

```
Train on 7982 samples, validate on 1000 samples
Epoch 1/9
7982/7982 [==============================] - 1s 125us/step - loss: 2.9912 - acc:
0.3419 - val_loss: 2.2328 - val_acc: 0.3540
Epoch 2/9
7982/7982 [==============================] - 1s 85us/step - loss: 1.8512 - acc: 0.
3514 - val_loss: 1.6510 - val_acc: 0.3540
Epoch 3/9
7982/7982 [==============================] - 1s 85us/step - loss: 1.3002 - acc: 0.
3578 - val_loss: 1.3110 - val_acc: 0.4420
Epoch 4/9
7982/7982 [==============================] - 1s 85us/step - loss: 0.7868 - acc: 0.
7844 - val_loss: 0.9464 - val_acc: 0.7920
Epoch 5/9
7982/7982 [==============================] - 1s 86us/step - loss: 0.5124 - acc: 0.
8903 - val_loss: 0.8999 - val_acc: 0.8210
Epoch 6/9
7982/7982 [==============================] - 1s 86us/step - loss: 0.3969 - acc: 0.
9163 - val_loss: 0.8966 - val_acc: 0.8210
Epoch 7/9
7982/7982 [==============================] - 1s 87us/step - loss: 0.3018 - acc: 0.
9347 - val_loss: 0.9494 - val_acc: 0.8050
Epoch 8/9
7982/7982 [==============================] - 1s 85us/step - loss: 0.2624 - acc: 0.
9409 - val_loss: 0.8960 - val_acc: 0.8230
Epoch 9/9
7982/7982 [==============================] - 1s 85us/step - loss: 1.4963 - acc: 0.
6119 - val_loss: 3.8286 - val_acc: 0.0060
2246/2246 [==============================] - 0s 106us/step
```

In [23]:

```
results
```

Out[23]:

```
[3.828641414005419, 0.005342831700801425]
```