In [1]:

```
import tensorflow as tf
```

In [2]:

```
tf.__version__
```

Out[2]:

```
'2.0.0'
```

In [3]:

```
from tensorflow.keras import layers, optimizers, metrics, datasets, Sequential, models
```

In [4]:

```
import os
```

In [5]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [10]:

```
(x_train, y_train), (x_test, y_test) = datasets.reuters.load_data(num_words=10000)
```

In [11]:

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

Out[11]:

```
((8982,), (8982,), (2246,), (2246,))
```

In [12]:

```
y_train.max(), y_train.min()
```

Out[12]:

```
(45, 0)
```

In [13]:

```
x_train[0][:5]
```

Out[13]:

```
[1, 2, 2, 8, 43]
```

In [14]:

```
y_train[0]
```

Out[14]:

```
3
```

In [15]:

```
max([max(sequence) for sequence in x_train])
```

Out[15]:

```
9999
```

数字和单词映射表，索引减3，因为0，1，2为padding、start of sequence、unknown保留的索引

In [16]:

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

In [17]:

```
decoded_review
```

Out[17]:

```
'? ? ? said as a result of its december acquisition of space co it expects e
arnings per share in 1987 of 1 15 to 1 30 dlrs per share up from 70 cts in 1
986 the company said pretax net should rise to nine to 10 mln dlrs from six
mln dlrs in 1986 and rental operation revenues to 19 to 22 mln dlrs from 12
5 mln dlrs it said cash flow per share this year should be 2 50 to three dlr
s reuter 3'
```

In [18]:

```
import numpy as np
```

向量化

In [19]:

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

In [20]:

```
x_train = vectorize_sequences(x_train)
x_test = vectorize_sequences(x_test)
```

In [21]:

```
x_train.shape
```

Out[21]:

```
(8982, 10000)
```

In [22]:

```
x_train[0].shape
```

Out[22]:

```
(10000,)
```

In [23]:

```
x_train[0]
```

Out[23]:

```
array([0., 1., 1., ..., 0., 0., 0.])
```

In [24]:

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

In [25]:

```
y_train = label_to_onehot(y_train)
```

In [31]:

```
y_test = label_to_onehot(y_test)
```

In [32]:

```
# from keras.utils.np_utils import to_categorical
# y_train = to_categorical(y_train)
# y_test = to_categorical(y_test)
```

In [33]:

```
y_train[0]
```

Out[33]:

```
array([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.])
```

In [35]:

```
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 [36]:

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

留出验证集

In [37]:

```
x_val = x_train[:1000]
x_train = x_train[1000:]

y_val = y_train[:1000]
y_train = y_train[1000:]
```

In [38]:

```
history = model.fit(x_train, y_train, epochs=20, batch_size=512, validation_data=(x_val, y_
```

```
Train on 7982 samples, validate on 1000 samples
Epoch 1/20
7982/7982 [==============================] - 1s 142us/sample - loss: 2.5443
- acc: 0.5360 - val_loss: 1.7089 - val_acc: 0.6400
Epoch 2/20
7982/7982 [==============================] - 0s 39us/sample - loss: 1.3924 -
acc: 0.7080 - val_loss: 1.3074 - val_acc: 0.7020
Epoch 3/20
7982/7982 [==============================] - 0s 39us/sample - loss: 1.0284 -
acc: 0.7833 - val_loss: 1.1428 - val_acc: 0.7650
Epoch 4/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.8005 -
acc: 0.8305 - val_loss: 1.0325 - val_acc: 0.7890
Epoch 5/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.6324 -
acc: 0.8661 - val_loss: 0.9650 - val_acc: 0.7950
Epoch 6/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.4973 -
acc: 0.8946 - val_loss: 0.9315 - val_acc: 0.7990
Epoch 7/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.4043 -
acc: 0.9153 - val_loss: 0.8930 - val_acc: 0.8160
Epoch 8/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.3244 -
acc: 0.9291 - val_loss: 0.9437 - val_acc: 0.8070
Epoch 9/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.2720 -
acc: 0.9394 - val_loss: 0.9704 - val_acc: 0.7940
Epoch 10/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.2310 -
acc: 0.9473 - val_loss: 0.9045 - val_acc: 0.8180
Epoch 11/20
7982/7982 [==============================] - 0s 40us/sample - loss: 0.1994 -
acc: 0.9511 - val_loss: 0.9689 - val_acc: 0.8080
Epoch 12/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1785 -
acc: 0.9529 - val_loss: 0.9375 - val_acc: 0.8110
Epoch 13/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1605 -
acc: 0.9534 - val_loss: 0.9824 - val_acc: 0.8190
Epoch 14/20
7982/7982 [==============================] - 0s 40us/sample - loss: 0.1438 -
acc: 0.9567 - val_loss: 1.0592 - val_acc: 0.7950
Epoch 15/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1382 -
acc: 0.9559 - val_loss: 0.9810 - val_acc: 0.8030
Epoch 16/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1289 -
acc: 0.9549 - val_loss: 1.0275 - val_acc: 0.8090
Epoch 17/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1253 -
acc: 0.9558 - val_loss: 1.0566 - val_acc: 0.8110
Epoch 18/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1178 -
acc: 0.9575 - val_loss: 1.0519 - val_acc: 0.8070
Epoch 19/20
```

```
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1173 -
acc: 0.9572 - val_loss: 1.0590 - val_acc: 0.8100
Epoch 20/20
7982/7982 [==============================] - 0s 39us/sample - loss: 0.1112 -
acc: 0.9563 - val_loss: 1.0808 - val_acc: 0.8030
```

In [39]:

```
history_dict = history.history
history_dict.keys()
```

Out[39]:

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

绘图

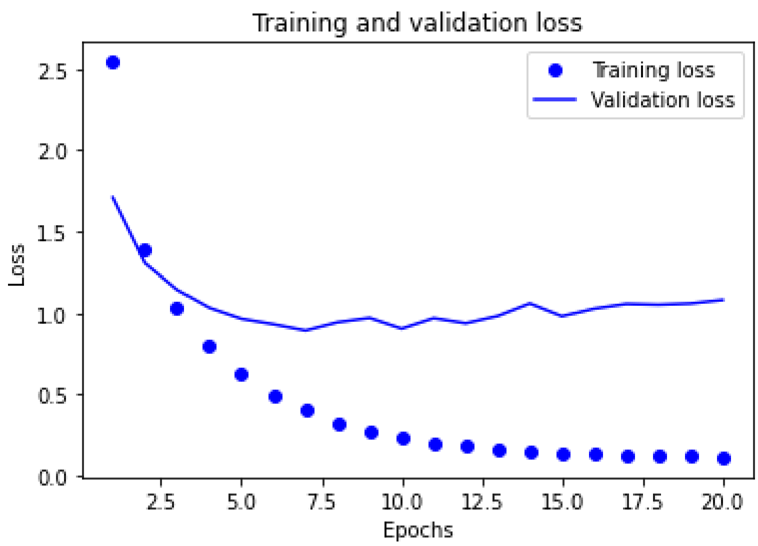
In [40]:

```
loss = history_dict['loss']
val_loss = history_dict['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()
```

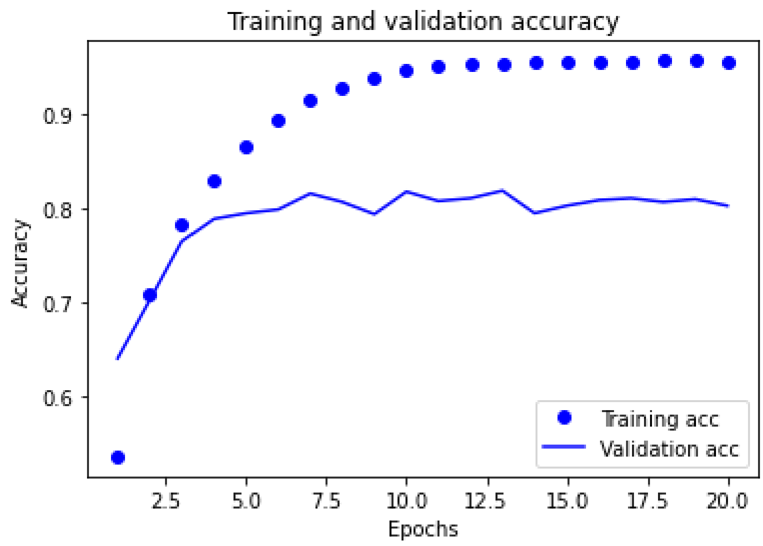


In [42]:

```
# plt.clf()     # 清除图像
acc = history_dict['acc']
val_acc = history_dict['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 [43]:

```
loss, acc = model.evaluate(x_test, y_test, verbose=0)
```

In [44]:

```
loss, acc
```

Out[44]:

```
(1.2142715910557436, 0.78762245)
```

In [45]:

```
model.predict(x_test)
```

Out[45]:

```
array([[4.3166124e-06, 2.0112477e-06, 1.4135250e-06, ..., 1.6920768e-07,
        6.1201211e-11, 2.5292979e-07],
       [5.3297165e-03, 4.1028306e-02, 1.3740624e-04, ..., 1.1357581e-07,
        1.0233130e-11, 2.1463749e-03],
       [4.4045858e-03, 8.6489767e-01, 7.9972480e-05, ..., 7.8108671e-05,
        3.6248016e-07, 6.6677942e-03],
       ...,
       [1.9819222e-06, 3.0813862e-05, 3.3245612e-07, ..., 2.0476131e-07,
        3.4353287e-09, 5.6891651e-07],
       [5.2254526e-03, 1.5864530e-01, 1.0250367e-02, ..., 5.2428310e-04,
        2.9391839e-04, 2.1144995e-03],
       [2.3016272e-04, 6.1111307e-01, 1.1649334e-03, ..., 1.6303831e-05,
        1.1194745e-05, 5.8803464e-05]], dtype=float32)
```