In [1]:
```
from keras.datasets import reuters
```

In [2]:
```
(train_data, train_labels), (test_data, test_labels) = reuters.load_data(
num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/reuters.npz (https://storag
e.googleapis.com/tensorflow/tf-keras-datasets/reuters.npz)
2110848/2110848 [==============================] - 1s 1us/step
```

In [3]:
```
 len(train_data)
```

Out[3]:
```
8982
```

In [4]:
```
len(test_data)
```

Out[4]:
```
2246
```

```
In [5]: train_data[10]

Out[5]: [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]
```

In [6]:
```
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]])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/reuters_word_index.json (htt
ps://storage.googleapis.com/tensorflow/tf-keras-datasets/reuters_word_index.json)
550378/550378 [==============================] - 0s 0us/step
```

In [7]:
```
train_labels[10]
```

Out[7]:
```
3
```

In [8]:
```
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 [9]:
```
def to_one_hot(labels, dimension=46):
    results = np.zeros((len(labels), dimension))
    for i, label in enumerate(labels):
        results[i, label] = 1.
    return results
one_hot_train_labels = to_one_hot(train_labels)
one_hot_test_labels = to_one_hot(test_labels)
```

In [10]:
```
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 [11]:
```
from keras import models
from keras import 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 [12]:
```
model.compile(optimizer='rmsprop',
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

In [13]:
```
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 [14]:
```
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/20
16/16 [==============================] - 3s 36ms/step - loss: 2.4585 - accuracy: 0.5281 - val_loss: 1.6231 -
val_accuracy: 0.6250
Epoch 2/20
16/16 [==============================] - 0s 12ms/step - loss: 1.3543 - accuracy: 0.6993 - val_loss: 1.2963 -
val_accuracy: 0.7130
Epoch 3/20
16/16 [==============================] - 0s 12ms/step - loss: 1.0275 - accuracy: 0.7795 - val_loss: 1.1554 -
val_accuracy: 0.7480
Epoch 4/20
16/16 [==============================] - 0s 12ms/step - loss: 0.8004 - accuracy: 0.8295 - val_loss: 1.0597 -
val_accuracy: 0.7660
Epoch 5/20
16/16 [==============================] - 0s 11ms/step - loss: 0.6405 - accuracy: 0.8603 - val_loss: 0.9774 -
val_accuracy: 0.7980
Epoch 6/20
16/16 [==============================] - 0s 12ms/step - loss: 0.5143 - accuracy: 0.8915 - val_loss: 0.9321 -
val_accuracy: 0.8110
Epoch 7/20
16/16 [==============================] - 0s 13ms/step - loss: 0.4089 - accuracy: 0.9131 - val_loss: 0.9229 -
val_accuracy: 0.8170
Epoch 8/20
16/16 [==============================] - 0s 13ms/step - loss: 0.3351 - accuracy: 0.9277 - val_loss: 0.9082 -
val_accuracy: 0.8160
Epoch 9/20
16/16 [==============================] - 0s 13ms/step - loss: 0.2743 - accuracy: 0.9387 - val_loss: 0.9248 -
val_accuracy: 0.8150
Epoch 10/20
16/16 [==============================] - 0s 16ms/step - loss: 0.2295 - accuracy: 0.9454 - val_loss: 0.9496 -
val_accuracy: 0.8080
Epoch 11/20
16/16 [==============================] - 0s 13ms/step - loss: 0.2032 - accuracy: 0.9495 - val_loss: 0.9882 -
val_accuracy: 0.7970
Epoch 12/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1777 - accuracy: 0.9521 - val_loss: 0.9972 -
val_accuracy: 0.8050
```

```
Epoch 13/20
16/16 [==============================] - 0s 11ms/step - loss: 0.1609 - accuracy: 0.9515 - val_loss: 0.9750 -
val_accuracy: 0.8110
Epoch 14/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1484 - accuracy: 0.9533 - val_loss: 0.9963 -
val_accuracy: 0.8100
Epoch 15/20
16/16 [==============================] - 0s 16ms/step - loss: 0.1352 - accuracy: 0.9551 - val_loss: 1.0606 -
val_accuracy: 0.8040
Epoch 16/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1283 - accuracy: 0.9570 - val_loss: 1.0428 -
val_accuracy: 0.8050
Epoch 17/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1232 - accuracy: 0.9565 - val_loss: 1.0595 -
val_accuracy: 0.8020
Epoch 18/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1160 - accuracy: 0.9575 - val_loss: 1.1345 -
val_accuracy: 0.8000
Epoch 19/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1154 - accuracy: 0.9578 - val_loss: 1.0652 -
val_accuracy: 0.8130
Epoch 20/20
16/16 [==============================] - 0s 12ms/step - loss: 0.1071 - accuracy: 0.9582 - val_loss: 1.1075 -
val_accuracy: 0.8050
```

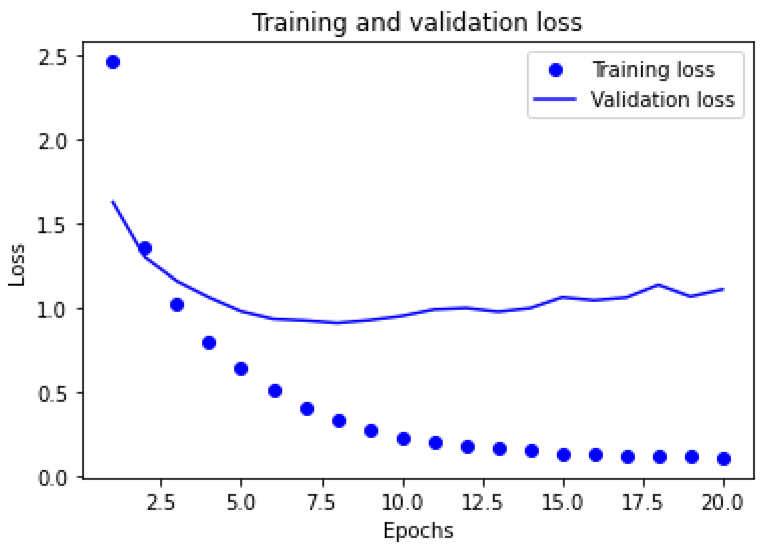
In [15]:
```python
import matplotlib.pyplot as plt

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()
```

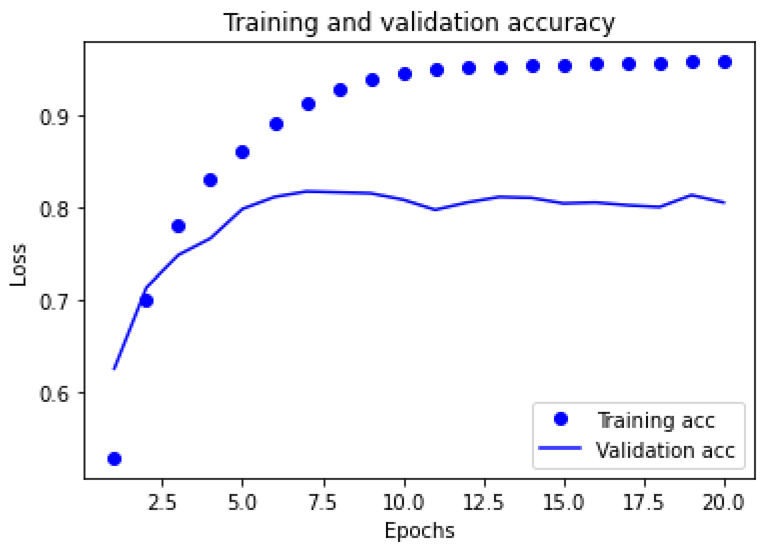


In [16]:
```
plt.clf()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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('Loss')
plt.legend()

plt.show()
```



```
In [17]: 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'))

         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)
```

```
Epoch 1/9
16/16 [==============================] - 1s 27ms/step - loss: 2.6932 - accuracy: 0.5193 - val_loss: 1.8354 -
val_accuracy: 0.6130
Epoch 2/9
16/16 [==============================] - 0s 11ms/step - loss: 1.5054 - accuracy: 0.6859 - val_loss: 1.3691 -
val_accuracy: 0.6970
Epoch 3/9
16/16 [==============================] - 0s 11ms/step - loss: 1.1023 - accuracy: 0.7606 - val_loss: 1.1711 -
val_accuracy: 0.7530
Epoch 4/9
16/16 [==============================] - 0s 11ms/step - loss: 0.8523 - accuracy: 0.8195 - val_loss: 1.0890 -
val_accuracy: 0.7510
Epoch 5/9
16/16 [==============================] - 0s 12ms/step - loss: 0.6734 - accuracy: 0.8588 - val_loss: 0.9902 -
val_accuracy: 0.7800
Epoch 6/9
16/16 [==============================] - 0s 12ms/step - loss: 0.5390 - accuracy: 0.8921 - val_loss: 0.9470 -
val_accuracy: 0.7970
Epoch 7/9
16/16 [==============================] - 0s 12ms/step - loss: 0.4364 - accuracy: 0.9132 - val_loss: 0.9102 -
val_accuracy: 0.8150
Epoch 8/9
16/16 [==============================] - 0s 12ms/step - loss: 0.3559 - accuracy: 0.9290 - val_loss: 0.9247 -
val_accuracy: 0.7980
Epoch 9/9
16/16 [==============================] - 0s 12ms/step - loss: 0.2931 - accuracy: 0.9396 - val_loss: 0.9122 -
```

```
val_accuracy: 0.8110
71/71 [==============================] - 0s 4ms/step - loss: 0.9865 - accuracy: 0.7925
```

```
In [18]: results
```

```
Out[18]: [0.9864797592163086, 0.7925200462341309]
```

```
In [19]: import copy
```

```
In [20]: test_labels_copy = copy.copy(test_labels)
```

```
In [21]:  np.random.shuffle(test_labels_copy)
```

```
In [22]: hits_array = np.array(test_labels) == np.array(test_labels_copy)
```

```
In [23]: float(np.sum(hits_array)) / len(test_labels)
```

```
Out[23]: 0.18788958147818344
```

```
In [24]: predictions = model.predict(x_test)
```

```
71/71 [==============================] - 0s 2ms/step
```

```
In [25]: predictions[0].shape
```

```
Out[25]: (46,)
```

```
In [26]: np.sum(predictions[0])
```

```
Out[26]: 1.0000001
```

```
In [27]: np.argmax(predictions[0])
```

```
Out[27]: 3
```

In [28]:
```
y_train = np.array(train_labels)
y_test = np.array(test_labels)
```

In [29]:
```
model.compile(optimizer='rmsprop',
            loss='sparse_categorical_crossentropy',
            metrics=['acc'])
```

```
In [30]: model = models.Sequential()
         model.add(layers.Dense(64, activation='relu', input_shape=(10000,)))
         model.add(layers.Dense(4, activation='relu'))
         model.add(layers.Dense(46, activation='softmax'))
         model.compile(optimizer='rmsprop',
         loss='categorical_crossentropy',
         metrics=['accuracy'])
         model.fit(partial_x_train,
         partial_y_train,
         epochs=20,
         batch_size=128,
         validation_data=(x_val, y_val))
```

```
Epoch 1/20
63/63 [==============================] - 1s 10ms/step - loss: 3.1057 - accuracy: 0.2442 - val_loss: 2.4535 - v
al_accuracy: 0.3180
Epoch 2/20
63/63 [==============================] - 0s 6ms/step - loss: 2.1014 - accuracy: 0.3583 - val_loss: 1.8753 - va
l_accuracy: 0.4710
Epoch 3/20
63/63 [==============================] - 0s 6ms/step - loss: 1.7030 - accuracy: 0.5079 - val_loss: 1.7067 - va
l_accuracy: 0.5350
Epoch 4/20
63/63 [==============================] - 0s 6ms/step - loss: 1.4523 - accuracy: 0.6616 - val_loss: 1.5097 - va
l_accuracy: 0.6550
Epoch 5/20
63/63 [==============================] - 0s 6ms/step - loss: 1.2689 - accuracy: 0.6907 - val_loss: 1.4406 - va
l_accuracy: 0.6570
Epoch 6/20
63/63 [==============================] - 0s 6ms/step - loss: 1.1581 - accuracy: 0.7031 - val_loss: 1.4048 - va
l_accuracy: 0.6620
Epoch 7/20
63/63 [==============================] - 0s 6ms/step - loss: 1.0798 - accuracy: 0.7162 - val_loss: 1.3995 - va
l_accuracy: 0.6730
Epoch 8/20
63/63 [==============================] - 0s 5ms/step - loss: 1.0134 - accuracy: 0.7316 - val_loss: 1.4029 - va
l_accuracy: 0.6790
Epoch 9/20
63/63 [==============================] - 0s 6ms/step - loss: 0.9597 - accuracy: 0.7428 - val_loss: 1.4255 - va
l_accuracy: 0.6790
Epoch 10/20
63/63 [==============================] - 0s 6ms/step - loss: 0.9124 - accuracy: 0.7593 - val_loss: 1.4682 - va
```

```
l_accuracy: 0.6870
Epoch 11/20
63/63 [==============================] - 0s 6ms/step - loss: 0.8735 - accuracy: 0.7730 - val_loss: 1.4902 - va
l_accuracy: 0.6830
Epoch 12/20
63/63 [==============================] - 0s 6ms/step - loss: 0.8378 - accuracy: 0.7829 - val_loss: 1.5443 - va
l_accuracy: 0.6740
Epoch 13/20
63/63 [==============================] - 0s 6ms/step - loss: 0.8053 - accuracy: 0.7899 - val_loss: 1.5904 - va
l_accuracy: 0.6780
Epoch 14/20
63/63 [==============================] - 0s 6ms/step - loss: 0.7780 - accuracy: 0.7942 - val_loss: 1.6225 - va
l_accuracy: 0.6840
Epoch 15/20
63/63 [==============================] - 0s 6ms/step - loss: 0.7529 - accuracy: 0.7982 - val_loss: 1.7090 - va
l_accuracy: 0.6820
Epoch 16/20
63/63 [==============================] - 0s 6ms/step - loss: 0.7292 - accuracy: 0.8047 - val_loss: 1.7221 - va
l_accuracy: 0.6830
Epoch 17/20
63/63 [==============================] - 0s 6ms/step - loss: 0.7089 - accuracy: 0.8094 - val_loss: 1.7522 - va
l_accuracy: 0.6900
Epoch 18/20
63/63 [==============================] - 0s 6ms/step - loss: 0.6891 - accuracy: 0.8136 - val_loss: 1.8107 - va
l_accuracy: 0.6900
Epoch 19/20
63/63 [==============================] - 0s 6ms/step - loss: 0.6720 - accuracy: 0.8156 - val_loss: 1.9451 - va
l_accuracy: 0.6880
Epoch 20/20
63/63 [==============================] - 0s 6ms/step - loss: 0.6538 - accuracy: 0.8211 - val_loss: 1.9806 - va
l_accuracy: 0.6950
```

```
Out[30]: <keras.callbacks.History at 0x20c5da67d30>
```

```
In [ ]:
```