In [31]:
```
from keras.datasets import imdb
import numpy as np
from keras import models
from keras import layers
from keras import optimizers
from keras import losses
from keras import metrics
import matplotlib.pyplot as plt

(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_word
s = 10000)
```

In [9]:
```
word_index = imdb.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 train_data[0]])
```

In [10]:
```
decoded_review
```

Out[10]:
```
"? this film was just brilliant casting location scenery story direction ever
yone's really suited the part they played and you could just imagine being th
ere robert ? is an amazing actor and now the same being director ? father cam
e from the same scottish island as myself so i loved the fact there was a rea
l connection with this film the witty remarks throughout the film were great
it was just brilliant so much that i bought the film as soon as it was releas
ed for ? and would recommend it to everyone to watch and the fly fishing was
amazing really cried at the end it was so sad and you know what they say if y
ou cry at a film it must have been good and this definitely was also ? to the
two little boy's that played the ? of norman and paul they were just brillian
t children are often left out of the ? list i think because the stars that pl
ay them all grown up are such a big profile for the whole film but these chil
dren are amazing and should be praised for what they have done don't you thin
k the whole story was so lovely because it was true and was someone's life af
ter all that was shared with us all"
```

In [15]:
```
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)
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

In [26]:
```
model = models.Sequential()
model.add(layers.Dense(16, activation = 'relu', input_shape = (10000,)))
model.add(layers.Dense(16, activation = 'relu'))
model.add(layers.Dense(1, activation = 'sigmoid'))
```

In [28]:

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [29]:

```
model.compile(optimizer = optimizers.RMSprop(lr = 0.001),
              loss = losses.binary_crossentropy,
              metrics = [metrics.binary_accuracy])

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

```
Epoch 1/20
30/30 [==============================] - 1s 38ms/step - loss: 0.5152 - binary
_accuracy: 0.7899 - val_loss: 0.4013 - val_binary_accuracy: 0.8557
Epoch 2/20
30/30 [==============================] - 1s 28ms/step - loss: 0.3117 - binary
_accuracy: 0.9019 - val_loss: 0.3080 - val_binary_accuracy: 0.8869
Epoch 3/20
30/30 [==============================] - 1s 26ms/step - loss: 0.2277 - binary
_accuracy: 0.9248 - val_loss: 0.2878 - val_binary_accuracy: 0.8852
Epoch 4/20
30/30 [==============================] - 1s 27ms/step - loss: 0.1813 - binary
_accuracy: 0.9385 - val_loss: 0.2807 - val_binary_accuracy: 0.8874
Epoch 5/20
30/30 [==============================] - 1s 27ms/step - loss: 0.1484 - binary
_accuracy: 0.9520 - val_loss: 0.2817 - val_binary_accuracy: 0.8890
Epoch 6/20
30/30 [==============================] - 1s 26ms/step - loss: 0.1216 - binary
_accuracy: 0.9626 - val_loss: 0.2893 - val_binary_accuracy: 0.8878
Epoch 7/20
30/30 [==============================] - 1s 26ms/step - loss: 0.1037 - binary
_accuracy: 0.9679 - val_loss: 0.3135 - val_binary_accuracy: 0.8795
Epoch 8/20
30/30 [==============================] - 1s 31ms/step - loss: 0.0840 - binary
_accuracy: 0.9765 - val_loss: 0.3457 - val_binary_accuracy: 0.8783
Epoch 9/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0708 - binary
_accuracy: 0.9807 - val_loss: 0.3453 - val_binary_accuracy: 0.8818
Epoch 10/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0594 - binary
_accuracy: 0.9849 - val_loss: 0.3756 - val_binary_accuracy: 0.8751
Epoch 11/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0450 - binary
_accuracy: 0.9903 - val_loss: 0.4288 - val_binary_accuracy: 0.8723
Epoch 12/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0389 - binary
_accuracy: 0.9917 - val_loss: 0.4405 - val_binary_accuracy: 0.8703
Epoch 13/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0282 - binary
_accuracy: 0.9953 - val_loss: 0.4794 - val_binary_accuracy: 0.8649
Epoch 14/20
30/30 [==============================] - 1s 33ms/step - loss: 0.0257 - binary
_accuracy: 0.9956 - val_loss: 0.4947 - val_binary_accuracy: 0.8670
Epoch 15/20
30/30 [==============================] - 1s 34ms/step - loss: 0.0197 - binary
_accuracy: 0.9971 - val_loss: 0.5293 - val_binary_accuracy: 0.8662
Epoch 16/20
30/30 [==============================] - 1s 35ms/step - loss: 0.0137 - binary
_accuracy: 0.9986 - val_loss: 0.5512 - val_binary_accuracy: 0.8685
Epoch 17/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0116 - binary
_accuracy: 0.9985 - val_loss: 0.5904 - val_binary_accuracy: 0.8675
Epoch 18/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0110 - binary
_accuracy: 0.9982 - val_loss: 0.6284 - val_binary_accuracy: 0.8670
Epoch 19/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0049 - binary
_accuracy: 0.9999 - val_loss: 0.6613 - val_binary_accuracy: 0.8670
```

```
Epoch 20/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0064 - binary
_accuracy: 0.9994 - val_loss: 0.6934 - val_binary_accuracy: 0.8661
```

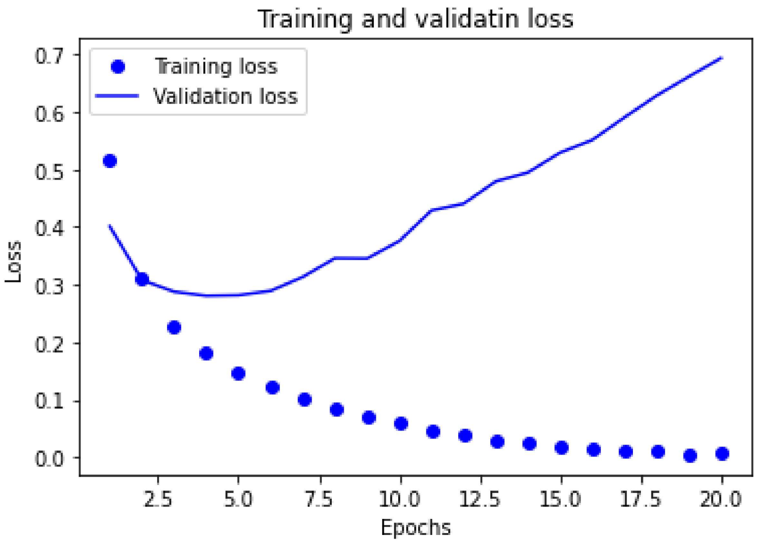
In [41]:

```
history_dict = history.history
loss_values = history_dict['loss']
binary_accuracy_values = history_dict['binary_accuracy']
val_binary_accuracy_values = history_dict['val_binary_accuracy']
val_loss_values = history_dict['val_loss']
#print(history_dict['binary_accuracy'])
#print(history_dict.keys())

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

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

plt.show()
```

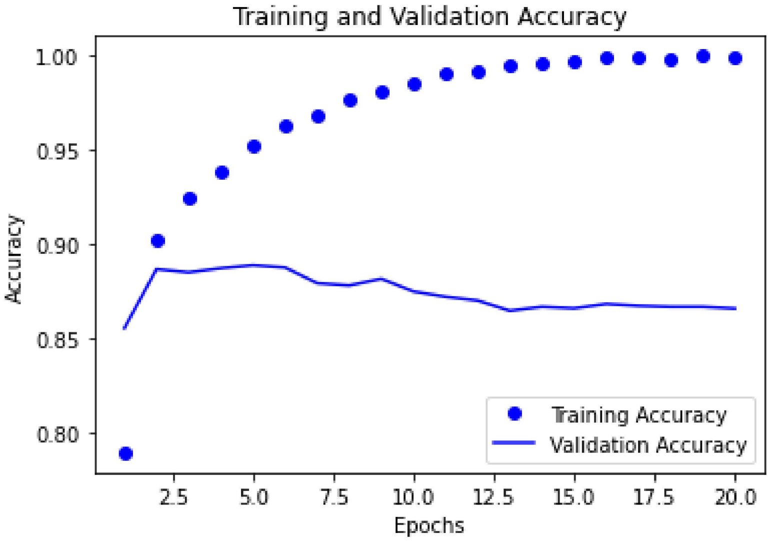


In [44]:

```
plt.clf()

plt.plot(epochs, binary_accuracy_values, 'bo', label = 'Training Accuracy')
plt.plot(epochs, val_binary_accuracy_values, 'b', label = 'Validation Accurac
y')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.show()
```



In [45]:

```
model2 = models.Sequential()
model2.add(layers.Dense(16, activation = 'relu', input_shape = (10000,)))
model2.add(layers.Dense(16, activation = 'relu'))
model2.add(layers.Dense(1, activation = 'sigmoid'))

model2.compile(optimizer = 'rmsprop',
    loss = 'binary_crossentropy',
    metrics = ['accuracy'])

model2.fit(x_train, y_train, epochs = 4, batch_size = 512)

results = model2.evaluate(x_test, y_test)
```

```
Epoch 1/4
49/49 [==============================] - 0s 9ms/step - loss: 0.4404 - accurac
y: 0.8273
Epoch 2/4
49/49 [==============================] - 0s 8ms/step - loss: 0.2554 - accurac
y: 0.9102
Epoch 3/4
49/49 [==============================] - 0s 8ms/step - loss: 0.1969 - accurac
y: 0.9300
Epoch 4/4
49/49 [==============================] - 0s 7ms/step - loss: 0.1667 - accurac
y: 0.9406
782/782 [==============================] - 1s 2ms/step - loss: 0.2933 - accur
acy: 0.8840
```

In [46]:

```
print(results)
```

```
[0.2932710647583008, 0.8839600086212158]
```

In [ ]: