In [2]:
```
import pandas as pd
import numpy as np
```

In [7]:
```
from keras.datasets import reuters

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

In [8]:
```
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
550378/550378 [==============================] - 1s 2us/step
```

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]:
```
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)

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 [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 = 9,
                    batch_size = 512,
                    validation_data=(x_val, y_val))
```

```
Epoch 1/9
16/16 [==============================] - 8s 252ms/step - loss: 2.7392 - accuracy:
0.5252 - val_loss: 1.7766 - val_accuracy: 0.6560
Epoch 2/9
16/16 [==============================] - 2s 144ms/step - loss: 1.4189 - accuracy:
0.7130 - val_loss: 1.3000 - val_accuracy: 0.7150
Epoch 3/9
16/16 [==============================] - 2s 145ms/step - loss: 1.0338 - accuracy:
0.7868 - val_loss: 1.1799 - val_accuracy: 0.7280
Epoch 4/9
16/16 [==============================] - 3s 164ms/step - loss: 0.8161 - accuracy:
0.8262 - val_loss: 1.0309 - val_accuracy: 0.7820
Epoch 5/9
16/16 [==============================] - 2s 137ms/step - loss: 0.6493 - accuracy:
0.8628 - val_loss: 0.9577 - val_accuracy: 0.7950
Epoch 6/9
16/16 [==============================] - 2s 153ms/step - loss: 0.5227 - accuracy:
0.8890 - val_loss: 0.9345 - val_accuracy: 0.7970
Epoch 7/9
16/16 [==============================] - 2s 142ms/step - loss: 0.4198 - accuracy:
0.9141 - val_loss: 0.9212 - val_accuracy: 0.8020
Epoch 8/9
16/16 [==============================] - 2s 153ms/step - loss: 0.3461 - accuracy:
0.9247 - val_loss: 0.8753 - val_accuracy: 0.8070
Epoch 9/9
16/16 [==============================] - 2s 140ms/step - loss: 0.2855 - accuracy:
0.9374 - val_loss: 0.9362 - val_accuracy: 0.8070
```

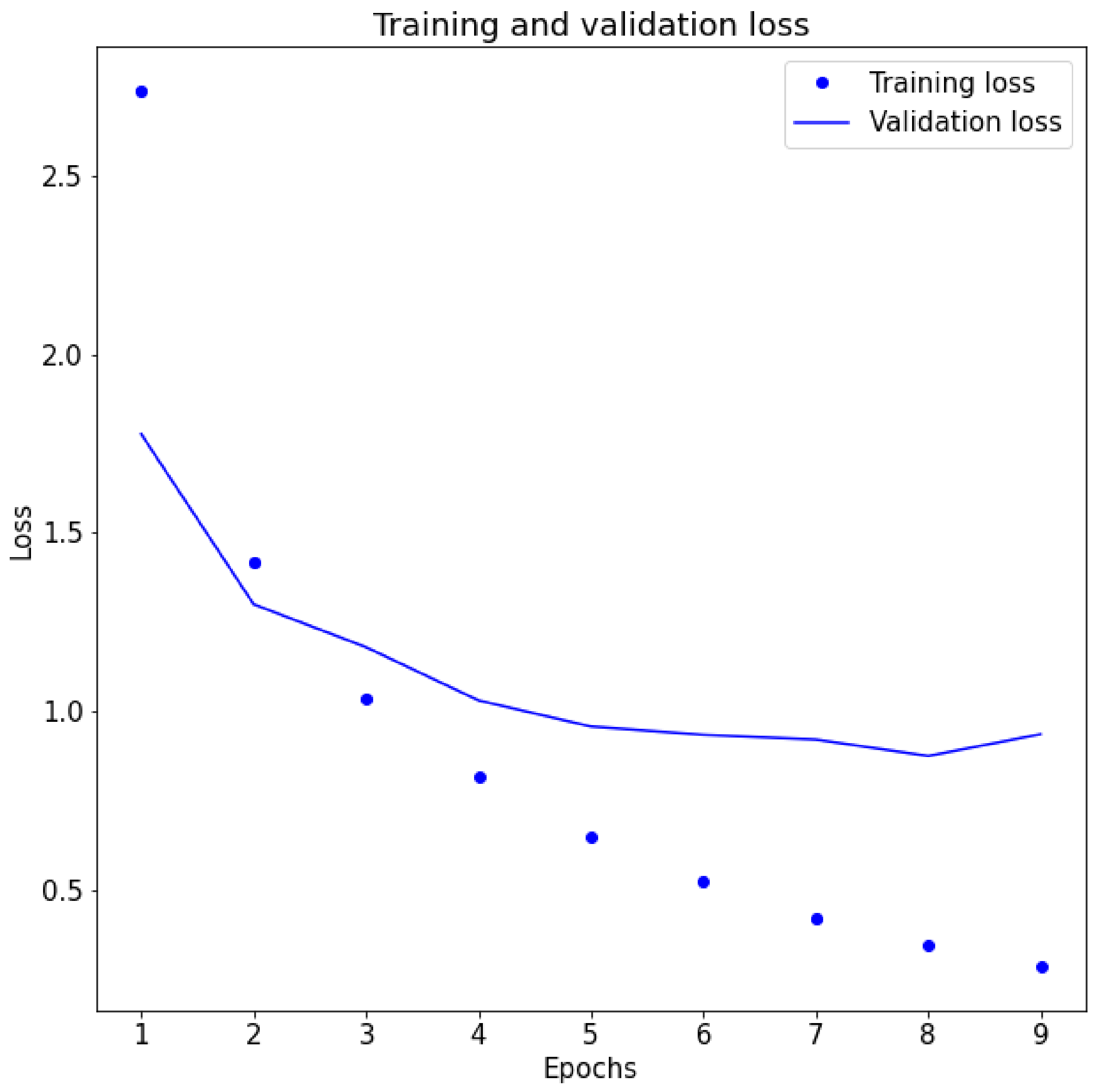
In [21]:

```
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.rc('font', size = 15)
plt.rc('figure', figsize=[10,10])
plt.show()
```

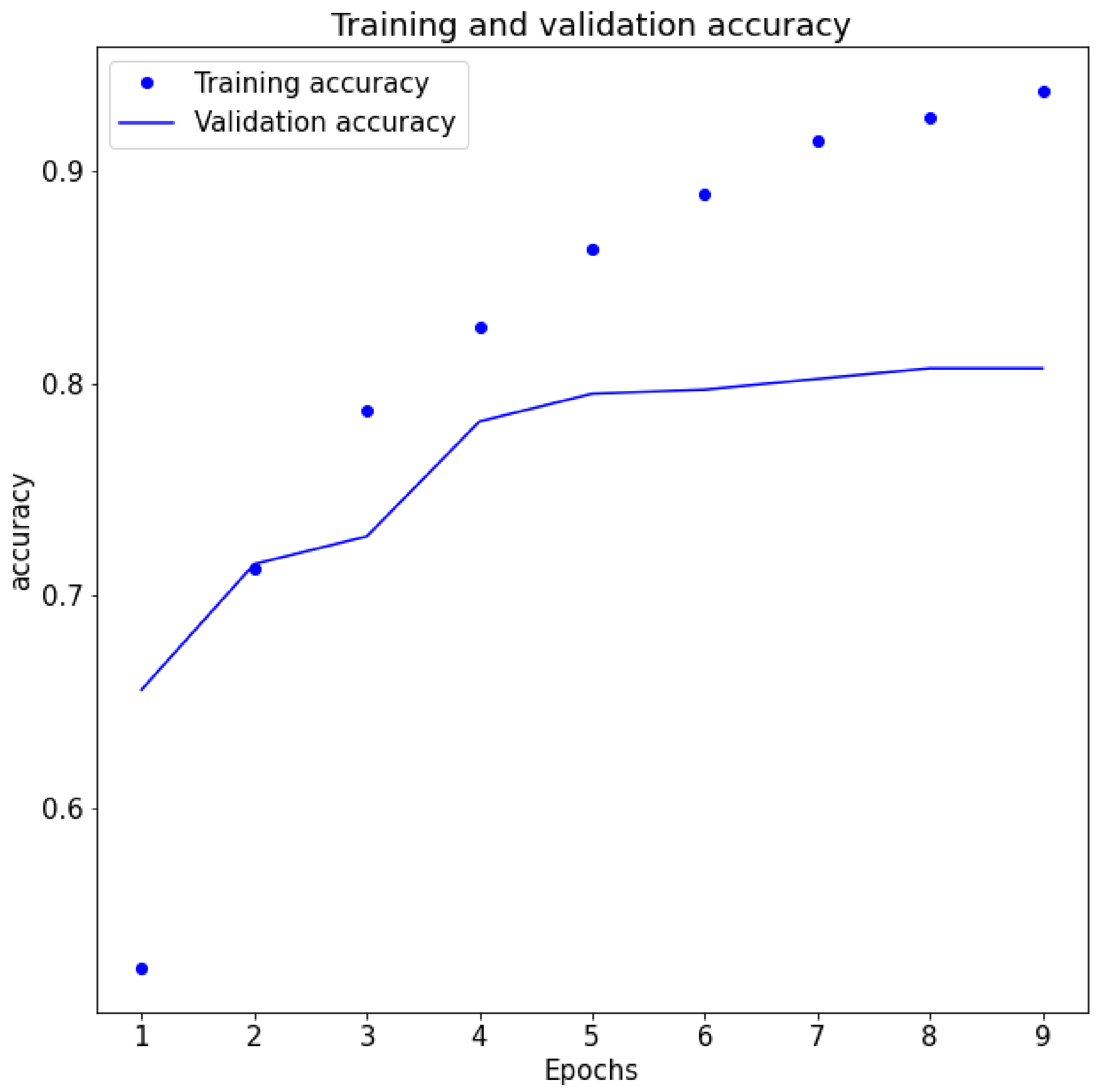


```
In [22]: acc = history.history['accuracy']
         val_acc = history.history['val_accuracy']

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

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

         plt.legend()
         plt.rc('font', size = 15)
         plt.rc('figure', figsize=[10,10])
         plt.show()
         results = model.evaluate(x_test, one_hot_test_labels)
         results
```



```
71/71 [==============================] - 1s 8ms/step - loss: 1.0345 - accuracy: 0.
7769
```

Out[22]:

```
[1.0344626903533936, 0.7769367694854736]
```

In [23]:

```
predictions = model.predict(x_test)
print(predictions[0].shape,'***',
np.sum(predictions[0]),'***',
np.argmax(predictions[0]))
```

```
71/71 [==============================] - 1s 9ms/step
(46,) *** 0.99999994 *** 3
```

In [ ]: