In [17]:
```
from keras.datasets import imdb
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras import losses
from keras import optimizers
import matplotlib.pyplot as plt
from keras import metrics
```

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

In [3]:
```
print(train_data.shape)
```

```
(25000,)
```

In [4]:
```
def vectorized_sequences(seq, dimension=10000):
    res = np.zeros((len(seq), dimension))

    for i, seq in enumerate(seq):
        res[i, seq] = 1
    return res
```

In [5]:
```
# prepairing data
x_train = vectorized_sequences(train_data)
x_test = vectorized_sequences(test_data)

y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

In [6]:
```
# create network
model = Sequential()
model.add(Dense(16, activation='relu', input_shape=(10000,)))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
WARNING:tensorflow:From /home/nahid/anaconda3/lib/python2.7/site-packages/tensorflo
w/python/framework/op_def_library.py:263: colocate_with (from tensorflow.python.fra
mework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [10]:
```
# compile

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

model.compile(optimizer='rmsprop',
    loss='binary_crossentropy',
    metrics=['acc'])
history = model.fit(partial_x_train,
    partial_y_train,
    epochs=20,
    batch_size=128,
    validation_data=(x_val, y_val))
```

```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 5s 337us/step - loss: 0.3961 - acc:
0.8413 - val_loss: 0.2826 - val_acc: 0.8879
Epoch 2/20
15000/15000 [==============================] - 1s 76us/step - loss: 0.2096 - acc:
0.9203 - val_loss: 0.2758 - val_acc: 0.8893
Epoch 3/20
15000/15000 [==============================] - 1s 78us/step - loss: 0.1509 - acc:
0.9468 - val_loss: 0.3029 - val_acc: 0.8870
Epoch 4/20
15000/15000 [==============================] - 1s 77us/step - loss: 0.1161 - acc:
0.9589 - val_loss: 0.3435 - val_acc: 0.8827
Epoch 5/20
15000/15000 [==============================] - 1s 78us/step - loss: 0.0871 - acc:
0.9697 - val_loss: 0.3796 - val_acc: 0.8787
Epoch 6/20
15000/15000 [==============================] - 1s 78us/step - loss: 0.0662 - acc:
0.9775 - val_loss: 0.4554 - val_acc: 0.8721
Epoch 7/20
15000/15000 [==============================] - 1s 77us/step - loss: 0.0484 - acc:
0.9825 - val_loss: 0.5028 - val_acc: 0.8698
Epoch 8/20
15000/15000 [==============================] - 1s 80us/step - loss: 0.0345 - acc:
0.9882 - val_loss: 0.6423 - val_acc: 0.8601
Epoch 9/20
15000/15000 [==============================] - 1s 78us/step - loss: 0.0239 - acc:
0.9923 - val_loss: 0.6817 - val_acc: 0.8577
Epoch 10/20
15000/15000 [==============================] - 1s 88us/step - loss: 0.0158 - acc:
0.9956 - val_loss: 0.7162 - val_acc: 0.8647
Epoch 11/20
15000/15000 [==============================] - 1s 82us/step - loss: 0.0094 - acc:
0.9978 - val_loss: 0.8632 - val_acc: 0.8563
Epoch 12/20
15000/15000 [==============================] - 1s 86us/step - loss: 0.0062 - acc:
0.9981 - val_loss: 0.9259 - val_acc: 0.8574
Epoch 13/20
15000/15000 [==============================] - 1s 82us/step - loss: 0.0031 - acc:
0.9993 - val_loss: 0.9693 - val_acc: 0.8610
Epoch 14/20
15000/15000 [==============================] - 1s 80us/step - loss: 0.0023 - acc:
0.9997 - val_loss: 1.0531 - val_acc: 0.8618
Epoch 15/20
15000/15000 [==============================] - 1s 79us/step - loss: 9.1765e-04 - ac
c: 0.9999 - val_loss: 1.1133 - val_acc: 0.8613
Epoch 16/20
15000/15000 [==============================] - 1s 83us/step - loss: 4.9944e-04 - ac
c: 0.9999 - val_loss: 1.1907 - val_acc: 0.8610
Epoch 17/20
15000/15000 [==============================] - 1s 83us/step - loss: 3.1179e-04 - ac
c: 1.0000 - val_loss: 1.2402 - val_acc: 0.8588
Epoch 18/20
15000/15000 [==============================] - 1s 91us/step - loss: 1.1886e-04 - ac
c: 1.0000 - val_loss: 1.3064 - val_acc: 0.8578
Epoch 19/20
15000/15000 [==============================] - 1s 92us/step - loss: 9.9862e-05 - ac
c: 1.0000 - val_loss: 1.3477 - val_acc: 0.8591
```

```
Epoch 20/20
15000/15000 [==============================] - 2s 120us/step - loss: 2.5465e-05 - a
cc: 1.0000 - val_loss: 1.4057 - val_acc: 0.8571
```

In [16]:
```
history_dict = history.history
history_dict.keys()
```

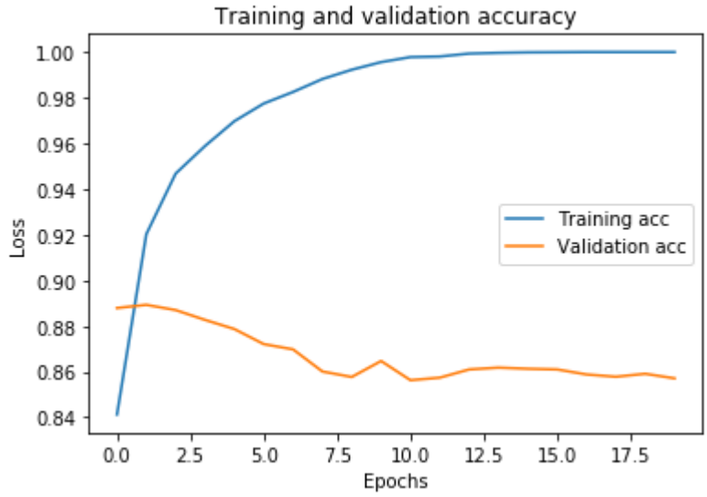
Out[16]:
```
['acc', 'loss', 'val_acc', 'val_loss']
```

In [20]:
```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
plt.plot(loss_values, label='Training loss')
plt.plot(val_loss_values, label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

In [21]:
```
plt.clf()
```

```
<Figure size 432x288 with 0 Axes>
```

In [23]:
```
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
plt.plot(acc_values, label='Training acc')
plt.plot(val_acc_values, label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [29]:
```
from keras.models import model_from_yaml
```

In [30]:
```
# serialize model to YAML
model_yaml = model.to_yaml()
with open("model.yaml", "w") as yaml_file:
    yaml_file.write(model_yaml)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

In [31]:
```
# load YAML and create model
yaml_file = open('model.yaml', 'r')
loaded_model_yaml = yaml_file.read()
yaml_file.close()
loaded_model = model_from_yaml(loaded_model_yaml)
# load weights into new model
loaded_model.load_weights("model.h5")
print("Loaded model from disk")

# evaluate loaded model on test data
loaded_model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accur
score = loaded_model.evaluate(x_test, y_test, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
/home/nahid/anaconda3/lib/python2.7/site-packages/keras/engine/saving.py:473: YAMLL
oadWarning: calling yaml.load() without Loader=... is deprecated, as the default Lo
ader is unsafe. Please read https://msg.pyyaml.org/load (https://msg.pyyaml.org/loa
d) for full details.
  config = yaml.load(yaml_string)

Loaded model from disk
acc: 84.04%
```

In [ ]:
```

```