In [43]:
```
from keras.datasets import imdb
```

In [44]:
```
(train_data,train_labels),(test_data,test_labels)=imdb.load_data(
num_words=10000)
```

In [8]:
```
train_data[0]
```

Out[8]:
```
[1,
 14,
 22,
 16,
 43,
 530,
 973,
 1622,
 1385,
 65,
 458,
 4468,
 66,
 3941,
 4,
 173,
 36,
 256,
 5,
 25,
 100,
 43,
 838,
 112,
 50,
 670,
 2,
 9,
 35,
 480,
 284,
 5,
 150,
 4,
 172,
 112,
 167,
 2,
 336,
 385,
 39,
 4,
 172,
 4536,
 1111,
 17,
 546,
 38,
 13,
 447,
 4,
 192,
 50,
 16,
 6,
 147,
 2025,
```

```
19,
14,
22,
4,
1920,
4613,
469,
4,
22,
71,
87,
12,
16,
43,
530,
38,
76,
15,
13,
1247,
4,
22,
17,
515,
17,
12,
16,
626,
18,
2,
5,
62,
386,
12,
8,
316,
8,
106,
5,
4,
2223,
5244,
16,
480,
66,
3785,
33,
4,
130,
12,
16,
38,
619,
5,
25,
124,
51,
```

```
36,
135,
48,
25,
1415,
33,
6,
22,
12,
215,
28,
77,
52,
5,
14,
407,
16,
82,
2,
8,
4,
107,
117,
5952,
15,
256,
4,
2,
7,
3766,
5,
723,
36,
71,
43,
530,
476,
26,
400,
317,
46,
7,
4,
2,
1029,
13,
104,
88,
4,
381,
15,
297,
98,
32,
2071,
56,
26,
```

```
 141,
 6,
 194,
 7486,
 18,
 4,
 226,
 22,
 21,
 134,
 476,
 26,
 480,
 5,
 144,
 30,
 5535,
 18,
 51,
 36,
 28,
 224,
 92,
 25,
 104,
 4,
 226,
 65,
 16,
 38,
 1334,
 88,
 12,
 16,
 283,
 5,
 16,
 4472,
 113,
 103,
 32,
 15,
 16,
 5345,
 19,
 178,
 32]
```

In [45]:
```
len(train_data[0])
```

Out[45]: 218

In [11]:
```
train_data[1]
```

Out[11]:
```
[1,
 194,
 1153,
 194,
 8255,
 78,
 228,
 5,
 6,
 1463,
 4369,
 5012,
 134,
 26,
 4,
 715,
 8,
 118,
 1634,
 14,
 394,
 20,
 13,
 119,
 954,
 189,
 102,
 5,
 207,
 110,
 3103,
 21,
 14,
 69,
 188,
 8,
 30,
 23,
 7,
 4,
 249,
 126,
 93,
 4,
 114,
 9,
 2300,
 1523,
 5,
 647,
 4,
 116,
 9,
 35,
 8163,
 4,
 229,
```

```
9,
340,
1322,
4,
118,
9,
4,
130,
4901,
19,
4,
1002,
5,
89,
29,
952,
46,
37,
4,
455,
9,
45,
43,
38,
1543,
1905,
398,
4,
1649,
26,
6853,
5,
163,
11,
3215,
2,
4,
1153,
9,
194,
775,
7,
8255,
2,
349,
2637,
148,
605,
2,
8003,
15,
123,
125,
68,
2,
6853,
15,
```

```
349,
165,
4362,
98,
5,
4,
228,
9,
43,
2,
1157,
15,
299,
120,
5,
120,
174,
11,
220,
175,
136,
50,
9,
4373,
228,
8255,
5,
2,
656,
245,
2350,
5,
4,
9837,
131,
152,
491,
18,
2,
32,
7464,
1212,
14,
9,
6,
371,
78,
22,
625,
64,
1382,
9,
8,
168,
145,
23,
4,
```

```
1690,
15,
16,
4,
1355,
5,
28,
6,
52,
154,
462,
33,
89,
78,
285,
16,
145,
95]
```

In [46]:
```
train_labels[0]
```

Out[46]:
```
1
```

In [48]:
```
max([max(sequence)for sequence in train_data])
```

Out[48]:
```
9999
```

In [49]:
```
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 [50]:
```
y_train=np.asarray(train_labels).astype('float32')
y_test=np.asarray(test_labels).astype('float32')
```

In [51]:
```
from keras import models
from keras import layers
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 [52]:
```
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
```

In [53]:
```
x_val=x_train[:10000]
partial_x_train=x_train[10000:]
```

In [54]:
```
y_val=y_train[:10000]
partial_y_train=y_train[10000:]
```

In [56]:
```
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['acc'])
history=model.fit(partial_x_train,
partial_y_train,
epochs=20,
batch_size=512,
validation_data=(x_val,y_val))
```

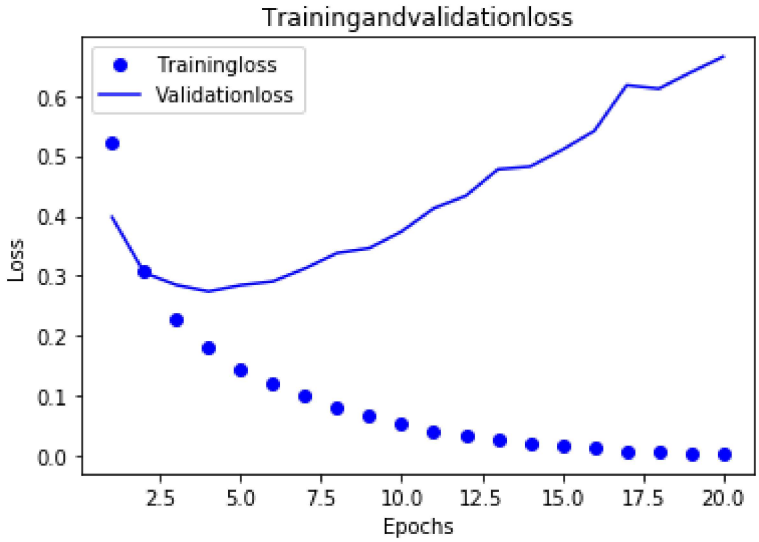
```
Train on 15000 samples, validate on 10000 samples
Epoch 1/20
15000/15000 [==============================] - 7s 455us/step - loss: 0.5206 -
acc: 0.7799 - val_loss: 0.3981 - val_acc: 0.8580
Epoch 2/20
15000/15000 [==============================] - 5s 315us/step - loss: 0.3095 -
acc: 0.9005 - val_loss: 0.3053 - val_acc: 0.8888
Epoch 3/20
15000/15000 [==============================] - 5s 312us/step - loss: 0.2278 -
acc: 0.9258 - val_loss: 0.2850 - val_acc: 0.8888
Epoch 4/20
15000/15000 [==============================] - 5s 310us/step - loss: 0.1800 -
acc: 0.9416 - val_loss: 0.2742 - val_acc: 0.8893
Epoch 5/20
15000/15000 [==============================] - 5s 321us/step - loss: 0.1448 -
acc: 0.9566 - val_loss: 0.2846 - val_acc: 0.8862
Epoch 6/20
15000/15000 [==============================] - 5s 318us/step - loss: 0.1197 -
acc: 0.9648 - val_loss: 0.2910 - val_acc: 0.8860
Epoch 7/20
15000/15000 [==============================] - 5s 305us/step - loss: 0.0994 -
acc: 0.9705 - val_loss: 0.3124 - val_acc: 0.8804
Epoch 8/20
15000/15000 [==============================] - 5s 316us/step - loss: 0.0825 -
acc: 0.9761 - val_loss: 0.3382 - val_acc: 0.8798
Epoch 9/20
15000/15000 [==============================] - 5s 322us/step - loss: 0.0670 -
acc: 0.9823 - val_loss: 0.3463 - val_acc: 0.8807
Epoch 10/20
15000/15000 [==============================] - 5s 302us/step - loss: 0.0536 -
acc: 0.9871 - val_loss: 0.3740 - val_acc: 0.8799
Epoch 11/20
15000/15000 [==============================] - 5s 309us/step - loss: 0.0418 -
acc: 0.9914 - val_loss: 0.4126 - val_acc: 0.8756
Epoch 12/20
15000/15000 [==============================] - 5s 311us/step - loss: 0.0338 -
acc: 0.9934 - val_loss: 0.4336 - val_acc: 0.8710
Epoch 13/20
15000/15000 [==============================] - 5s 314us/step - loss: 0.0285 -
acc: 0.9944 - val_loss: 0.4776 - val_acc: 0.8706
Epoch 14/20
15000/15000 [==============================] - 5s 311us/step - loss: 0.0203 -
acc: 0.9975 - val_loss: 0.4822 - val_acc: 0.8729
Epoch 15/20
15000/15000 [==============================] - 5s 324us/step - loss: 0.0160 -
acc: 0.9979 - val_loss: 0.5101 - val_acc: 0.8709
Epoch 16/20
15000/15000 [==============================] - 5s 320us/step - loss: 0.0140 -
acc: 0.9982 - val_loss: 0.5419 - val_acc: 0.8687
Epoch 17/20
15000/15000 [==============================] - 5s 307us/step - loss: 0.0082 -
acc: 0.9995 - val_loss: 0.6178 - val_acc: 0.8661
Epoch 18/20
15000/15000 [==============================] - 5s 309us/step - loss: 0.0068 -
acc: 0.9995 - val_loss: 0.6119 - val_acc: 0.8660
Epoch 19/20
15000/15000 [==============================] - 5s 335us/step - loss: 0.0056 -
```

```
acc: 0.9995 - val_loss: 0.6393 - val_acc: 0.8665
Epoch 20/20
15000/15000 [==============================] - 5s 307us/step - loss: 0.0043 -
acc: 0.9997 - val_loss: 0.6652 - val_acc: 0.8662
```

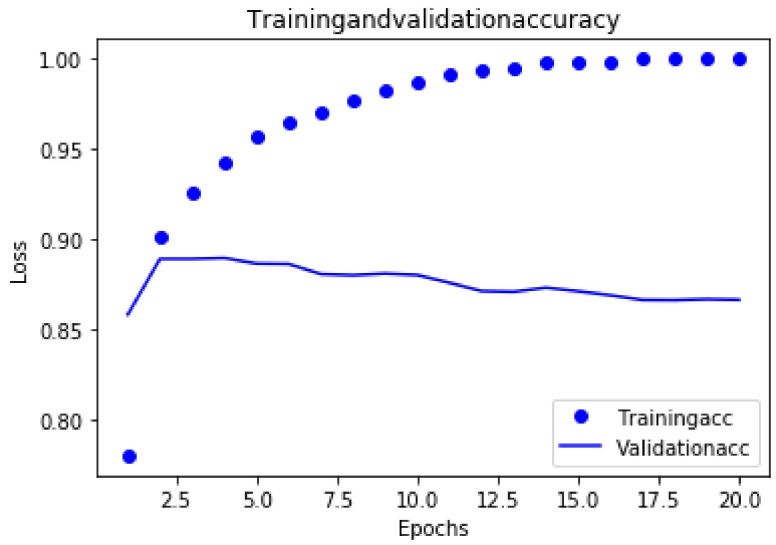
In [57]:
```
history_dict=history.history
history_dict.keys()
```

Out[57]:
```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

In [58]:
```
import matplotlib.pyplot as plt
history_dict=history.history
loss_values=history_dict['loss']
val_loss_values=history_dict['val_loss']
acc = history_dict['acc']
epochs = range(1, len(acc) + 1)
plt.plot(epochs,loss_values,'bo',label='Trainingloss')
plt.plot(epochs,val_loss_values,'b',label='Validationloss')
plt.title('Trainingandvalidationloss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [60]:
```
plt.clf()
val_acc_values=history_dict['val_acc']
plt.plot(epochs,acc,'bo',label='Trainingacc')
plt.plot(epochs,val_acc_values,'b',label='Validationacc')
plt.title('Trainingandvalidationaccuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [61]:
```
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'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(x_train,y_train,epochs=4,batch_size=512)
results=model.evaluate(x_test,y_test)
```

```
Epoch 1/4
25000/25000 [==============================] - 6s 233us/step - loss: 0.4582 -
acc: 0.8214
Epoch 2/4
25000/25000 [==============================] - 5s 199us/step - loss: 0.2656 -
acc: 0.9103
Epoch 3/4
25000/25000 [==============================] - 5s 194us/step - loss: 0.2028 -
acc: 0.9296
Epoch 4/4
25000/25000 [==============================] - 5s 198us/step - loss: 0.1706 -
acc: 0.9408
25000/25000 [==============================] - 7s 263us/step
```

In [62]:
```
results
```

Out[62]:
```
[0.2953490615367889, 0.88184]
```

In [63]:
```
model.predict(x_test)
```

Out[63]:
```
array([[0.2568012 ],
       [0.9981084 ],
       [0.9541164 ],
       ...,
       [0.14704984],
       [0.08720043],
       [0.6184923 ]], dtype=float32)
```

In [64]:
```
model=models.Sequential()
model.add(layers.Dense(16,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(x_train,y_train,epochs=4,batch_size=512)
results=model.evaluate(x_test,y_test)
```

```
Epoch 1/4
25000/25000 [==============================] - 6s 239us/step - loss: 0.4727 -
acc: 0.8175
Epoch 2/4
25000/25000 [==============================] - 5s 201us/step - loss: 0.2550 -
acc: 0.9074
Epoch 3/4
25000/25000 [==============================] - 5s 193us/step - loss: 0.1928 -
acc: 0.9308
Epoch 4/4
25000/25000 [==============================] - 5s 201us/step - loss: 0.1645 -
acc: 0.9422
25000/25000 [==============================] - 6s 246us/step
```

In [65]:
```
results
```

Out[65]:
```
[0.3185197500896454, 0.87608]
```

In [66]:
```
model.predict(x_test)
```

Out[66]:
```
array([[0.22329572],
       [0.9997761 ],
       [0.9865022 ],
       ...,
       [0.12615153],
       [0.07306567],
       [0.6919803 ]], dtype=float32)
```

In [67]:
```
model=models.Sequential()
model.add(layers.Dense(16,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(x_train,y_train,epochs=4,batch_size=512)
results=model.evaluate(x_test,y_test)
```

```
Epoch 1/4
25000/25000 [==============================] - 6s 229us/step - loss: 0.4442 -
acc: 0.8309
Epoch 2/4
25000/25000 [==============================] - 5s 197us/step - loss: 0.2770 -
acc: 0.9065
Epoch 3/4
25000/25000 [==============================] - 5s 193us/step - loss: 0.2179 -
acc: 0.9262
Epoch 4/4
25000/25000 [==============================] - 5s 193us/step - loss: 0.1868 -
acc: 0.9358 0s - loss: 0.1873 - acc:
25000/25000 [==============================] - 6s 240us/step
```

In [68]:
```
results
```

Out[68]:
```
[0.2812515107536316, 0.88756]
```

In [69]:
```
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'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(x_train,y_train,epochs=4,batch_size=512)
results=model.evaluate(x_test,y_test)
```

```
Epoch 1/4
25000/25000 [==============================] - 6s 231us/step - loss: 0.4431 -
acc: 0.8228
Epoch 2/4
25000/25000 [==============================] - 5s 193us/step - loss: 0.2506 -
acc: 0.9089
Epoch 3/4
25000/25000 [==============================] - 5s 194us/step - loss: 0.1941 -
acc: 0.9304
Epoch 4/4
25000/25000 [==============================] - 5s 193us/step - loss: 0.1597 -
acc: 0.9434
25000/25000 [==============================] - 6s 242us/step
```

In [70]:
```
results
```

Out[70]:
```
[0.3193111909008026, 0.87688]
```

In [71]:
```
model=models.Sequential()
model.add(layers.Dense(32,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(32,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(x_train,y_train,epochs=4,batch_size=512)
results=model.evaluate(x_test,y_test)
```

```
Epoch 1/4
25000/25000 [==============================] - 6s 254us/step - loss: 0.4410 -
acc: 0.8190
Epoch 2/4
25000/25000 [==============================] - 5s 208us/step - loss: 0.2470 -
acc: 0.9099
Epoch 3/4
25000/25000 [==============================] - 5s 203us/step - loss: 0.1900 -
acc: 0.9307
Epoch 4/4
25000/25000 [==============================] - 5s 217us/step - loss: 0.1538 -
acc: 0.9436
25000/25000 [==============================] - 7s 263us/step
```

In [72]:
```
results
```

Out[72]:
```
[0.30791939019680026, 0.88056]
```

In [73]:
```
model=models.Sequential()
model.add(layers.Dense(16,activation='relu',input_shape=(10000,)))
model.add(layers.Dense(64,activation='relu'))
model.add(layers.Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
model.fit(x_train,y_train,epochs=4,batch_size=512)
results=model.evaluate(x_test,y_test)
```

```
Epoch 1/4
25000/25000 [==============================] - 6s 240us/step - loss: 0.4426 -
acc: 0.8207
Epoch 2/4
25000/25000 [==============================] - 5s 195us/step - loss: 0.2423 -
acc: 0.9101
Epoch 3/4
25000/25000 [==============================] - 5s 203us/step - loss: 0.1908 -
acc: 0.9302
Epoch 4/4
25000/25000 [==============================] - 5s 195us/step - loss: 0.1578 -
acc: 0.9426
25000/25000 [==============================] - 6s 244us/step
```

In [74]:
```
results
```

Out[74]:
```
[0.3053156338882446, 0.88092]
```

In [ ]: