In [1]:
```
import os
import numpy as np
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense
from keras.utils.np_utils import to_categorical
```

```
Using TensorFlow backend.
```

In [2]:
```
#Read Data from Dataset
data_dir = 'D:/Deeplearning/datasets/bbc'
labels = []
texts = []
label_count = 0
for label_type in ['business', 'entertainment', 'politics', 'sport', 'tech']:
    dir_name = os.path.join(data_dir, label_type)
    for fname in os.listdir(dir_name):
        f = open(os.path.join(dir_name, fname), encoding="utf8", errors='ignor
e')
        texts.append(f.read())
        f.close()
        labels.append(label_count)
    label_count = label_count + 1
```

In [6]:
```
maxlen = 375 # Cut off after 375 words in tokenizer
test_samples = 100
training_samples = 1625
validation_samples = 500
max_words = 10000 # Size of dictionary for our problem
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
```

In [9]:
```
#Setting Parameters
maxlen = 375 # Cut off after 375 words in tokenizer
test_samples = 100
training_samples = 1625
validation_samples = 500
max_words = 10000 # Size of dictionary for our problem
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
```

In [11]:
```
len(sequences)
```

Out[11]:
```
2225
```

In [16]:
```
word_index = tokenizer.word_index
data = pad_sequences(sequences, maxlen=maxlen)
labels = np.asarray(labels)
```

In [17]:
```
data.shape
```

Out[17]:
```
(2225, 375)
```

In [18]:
```
# Randomly get training and validation samples
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
```

In [21]:
```
data.shape
```

Out[21]:
```
(2225, 375)
```

In [50]:
```
test_data = data[:100]
test_label = labels[:100]
x_train = data[100:training_samples+100]
y_train = labels[100:training_samples+100]
x_val = data[training_samples: training_samples + validation_samples]
y_val = labels[training_samples: training_samples + validation_samples]
```

In [51]:
```
test_data.shape
```

Out[51]:
```
(100, 375)
```

In [52]:
```
y_train.shape
```

Out[52]:
```
(1625,)
```

In [53]:
```
x_val.shape
```

Out[53]:
```
(500, 375)
```

In [54]:
```
y_val.shape
```

Out[54]:
```
(500,)
```

In [55]:
```
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
```

In [56]:
```
y_train.shape
```

Out[56]:
```
(1625, 5)
```

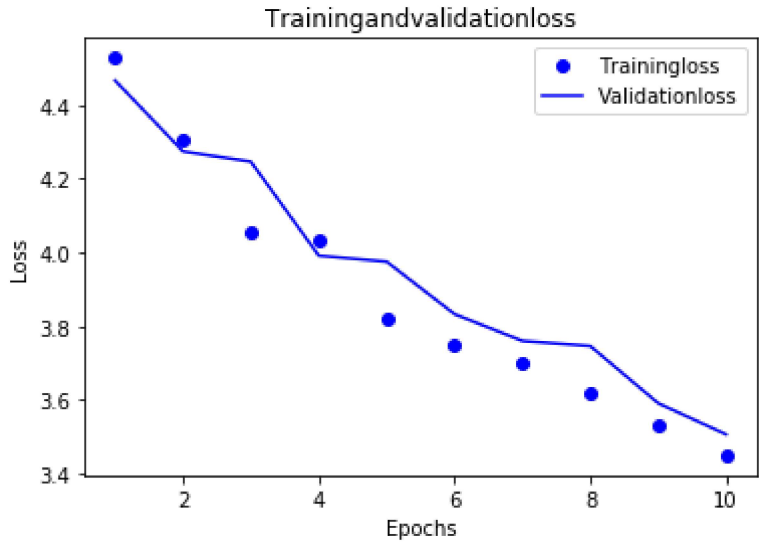
In [57]:
```
# Start creating the model
from keras import models
from keras import layers
model=models.Sequential()
model.add(layers.Dense(32,activation='relu',input_shape=(375,)))
model.add(layers.Dense(16,activation='relu'))
model.add(layers.Dense(5,activation='sigmoid'))
```

In [58]:
```
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['accuracy'])
```

In [60]:
```
history=model.fit(x_train,
y_train,
epochs=10,
batch_size=512,
validation_data=(x_val,y_val))
```

```
Train on 1625 samples, validate on 500 samples
Epoch 1/10
1625/1625 [==============================] - 0s 26us/step - loss: 4.5292 - ac
c: 0.7153 - val_loss: 4.4678 - val_acc: 0.7192
Epoch 2/10
1625/1625 [==============================] - 0s 23us/step - loss: 4.3078 - ac
c: 0.7300 - val_loss: 4.2750 - val_acc: 0.7328
Epoch 3/10
1625/1625 [==============================] - 0s 24us/step - loss: 4.0546 - ac
c: 0.7458 - val_loss: 4.2471 - val_acc: 0.7340
Epoch 4/10
1625/1625 [==============================] - 0s 30us/step - loss: 4.0356 - ac
c: 0.7471 - val_loss: 3.9916 - val_acc: 0.7496
Epoch 5/10
1625/1625 [==============================] - 0s 21us/step - loss: 3.8187 - ac
c: 0.7617 - val_loss: 3.9753 - val_acc: 0.7512
Epoch 6/10
1625/1625 [==============================] - 0s 19us/step - loss: 3.7472 - ac
c: 0.7664 - val_loss: 3.8329 - val_acc: 0.7612
Epoch 7/10
1625/1625 [==============================] - 0s 17us/step - loss: 3.6998 - ac
c: 0.7690 - val_loss: 3.7600 - val_acc: 0.7652
Epoch 8/10
1625/1625 [==============================] - 0s 21us/step - loss: 3.6176 - ac
c: 0.7745 - val_loss: 3.7461 - val_acc: 0.7660
Epoch 9/10
1625/1625 [==============================] - 0s 19us/step - loss: 3.5297 - ac
c: 0.7794 - val_loss: 3.5906 - val_acc: 0.7756
Epoch 10/10
1625/1625 [==============================] - 0s 19us/step - loss: 3.4491 - ac
c: 0.7852 - val_loss: 3.5060 - val_acc: 0.7816
```

In [61]:
```
import matplotlib.pyplot as plt
history_dict=history.history
loss_values=history_dict['loss']
acc = history_dict['acc']
val_loss_values=history_dict['val_loss']
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 [45]:
```
```

In [64]:
```
test_label.shape
```

Out[64]:
```
(100,)
```

In [66]:
```
one_hot_test_labels=to_categorical(test_label)
```

In [67]:
```
result = model.evaluate(test_data,one_hot_test_labels)
```

```
100/100 [==============================] - 0s 113us/step
```

In [68]:
```
result
```

Out[68]:
```
[3.4587248802185058, 0.7840000081062317]
```

In [ ]:
```
```