10.1 ¶

In [1]:
```
import string
import os
import numpy as np
```

In [2]:
```
with open('../../../data/external/imdb/aclImdb/train/neg/8731_1.txt') as f:
    x = f.read()
x
```

Out[2]:
```
'the lowest score possible is one star? that\'s a shame. really, i\'m going t
o lobby IMDb for a "zero stars" option. to give this film even a single star
is giving WAY too much. am i the only one who noticed the microphones danglin
g over hopper\'s head at the station? and the acting, or should i say the lac
k thereof? apparently talent wasn\'t a factor when the casting director came
to town. my little sister\'s elementary school talent show provides greater r
ange and depth of emotion. and those fake irish accents were like nails on a
chalk board. the only thing that could have made this movie worse would have
been...oh, wait, no,no, it\'s already as bad as it can get.'
```

In [3]:
```
def tokenize(sentence):
    tokens = []
    sentence = sentence.strip()
    sentence = sentence.translate(str.maketrans('','',string.punctuation))
    sentence = sentence.lower()
    tokens = sentence.split()
    # tokenize the sentence
    return tokens
y = tokenize(x)
```

In [4]:
```
def ngram(tokens, n):
    ngrams = []
    # Create ngrams
#     for i in range(n):
#         for w in range(len(tokens)):
#             tokens.pop
    for k in range(1,n+1):
        for i in range(len(tokens)-k+1):
            temp=[tokens[j] for j in range(i,i+k)]
            ngrams.append(" ".join(temp))

    return ngrams
ngram(y,3)
```

Out[4]:
```
['the',
 'lowest',
 'score',
 'possible',
 'is',
 'one',
 'star',
 'thats',
 'a',
 'shame',
 'really',
 'im',
 'going',
 'to',
 'lobby',
 'imdb',
 'for',
 'a',
 'zero',
 'stars',
 'option',
 'to',
 'give',
 'this',
 'film',
 'even',
 'a',
 'single',
 'star',
 'is',
 'giving',
 'way',
 'too',
 'much',
 'am',
 'i',
 'the',
 'only',
 'one',
 'who',
 'noticed',
 'the',
 'microphones',
 'dangling',
 'over',
 'hoppers',
 'head',
 'at',
 'the',
 'station',
 'and',
 'the',
 'acting',
 'or',
 'should',
 'i',
 'say',
 'the',
 'lack',
 'thereof',
 'apparently',
 'talent',
 'wasnt',
 'a',
 'factor',
 'when',
 'the',
 'casting',
 'director',
 'came',
 'to',
 'town',
 'my',
 'little',
 'sisters',
 'elementary',
 'school',
 'talent',
 'show',
 'provides',
 'greater',
 'range',
 'and',
 'depth',
 'of',
 'emotion',
 'and',
 'those',
 'fake',
 'irish',
 'accents',
 'were',
 'like',
 'nails',
 'on',
 'a',
 'chalk',
 'board',
 'the',
 'only',
 'thing',
 'that',
 'could',
 'have',
 'made',
 'this',
 'movie',
 'worse',
 'would',
 'have',
 'beenoh',
 'wait',
 'nono',
 'its',
 'already',
 'as',
 'bad',
 'as',
 'it',
 'can',
 'get',
 'the lowest',
 'lowest score',
 'score possible',
 'possible is',
 'is one',
 'one star',
 'star thats',
 'thats a',
 'a shame',
 'shame really',
 'really im',
 'im going',
 'going to',
 'to lobby',
 'lobby imdb',
 'imdb for',
 'for a',
 'a zero',
 'zero stars',
 'stars option',
 'option to',
 'to give',
 'give this',
 'this film',
 'film even',
 'even a',
 'a single',
 'single star',
 'star is',
 'is giving',
 'giving way',
 'way too',
 'too much',
 'much am',
 'am i',
 'i the',
 'the only',
 'only one',
 'one who',
 'who noticed',
 'noticed the',
 'the microphones',
 'microphones dangling',
 'dangling over',
 'over hoppers',
 'hoppers head',
 'head at',
 'at the',
 'the station',
 'station and',
 'and the',
 'the acting',
 'acting or',
 'or should',
 'should i',
 'i say',
 'say the',
 'the lack',
 'lack thereof',
 'thereof apparently',
 'apparently talent',
 'talent wasnt',
 'wasnt a',
 'a factor',
 'factor when',
 'when the',
 'the casting',
 'casting director',
 'director came',
 'came to',
 'to town',
 'town my',
 'my little',
 'little sisters',
 'sisters elementary',
 'elementary school',
 'school talent',
 'talent show',
 'show provides',
 'provides greater',
 'greater range',
 'range and',
 'and depth',
 'depth of',
 'of emotion',
 'emotion and',
 'and those',
 'those fake',
 'fake irish',
 'irish accents',
 'accents were',
 'were like',
 'like nails',
 'nails on',
 'on a',
 'a chalk',
 'chalk board',
 'board the',
 'the only',
 'only thing',
 'thing that',
 'that could',
 'could have',
 'have made',
 'made this',
 'this movie',
 'movie worse',
 'worse would',
 'would have',
 'have beenoh',
 'beenoh wait',
 'wait nono',
 'nono its',
 'its already',
 'already as',
 'as bad',
 'bad as',
 'as it',
 'it can',
 'can get',
 'the lowest score',
 'lowest score possible',
 'score possible is',
 'possible is one',
 'is one star',
 'one star thats',
 'star thats a',
 'thats a shame',
 'a shame really',
 'shame really im',
 'really im going',
 'im going to',
 'going to lobby',
 'to lobby imdb',
 'lobby imdb for',
 'imdb for a',
 'for a zero',
 'a zero stars',
 'zero stars option',
 'stars option to',
 'option to give',
 'to give this',
 'give this film',
 'this film even',
 'film even a',
 'even a single',
 'a single star',
 'single star is',
 'star is giving',
 'is giving way',
 'giving way too',
 'way too much',
 'too much am',
 'much am i',
 'am i the',
 'i the only',
 'the only one',
 'only one who',
 'one who noticed',
 'who noticed the',
 'noticed the microphones',
 'the microphones dangling',
 'microphones dangling over',
 'dangling over hoppers',
 'over hoppers head',
 'hoppers head at',
 'head at the',
 'at the station',
 'the station and',
 'station and the',
 'and the acting',
 'the acting or',
 'acting or should',
 'or should i',
 'should i say',
 'i say the',
 'say the lack',
 'the lack thereof',
 'lack thereof apparently',
 'thereof apparently talent',
 'apparently talent wasnt',
 'talent wasnt a',
 'wasnt a factor',
 'a factor when',
 'factor when the',
 'when the casting',
 'the casting director',
 'casting director came',
 'director came to',
 'came to town',
 'to town my',
 'town my little',
 'my little sisters',
 'little sisters elementary',
 'sisters elementary school',
 'elementary school talent',
 'school talent show',
 'talent show provides',
 'show provides greater',
 'provides greater range',
 'greater range and',
 'range and depth',
 'and depth of',
 'depth of emotion',
 'of emotion and',
 'emotion and those',
 'and those fake',
 'those fake irish',
 'fake irish accents',
 'irish accents were',
 'accents were like',
 'were like nails',
 'like nails on',
 'nails on a',
 'on a chalk',
 'a chalk board',
 'chalk board the',
 'board the only',
 'the only thing',
 'only thing that',
 'thing that could',
 'that could have',
 'could have made',
 'have made this',
 'made this movie',
 'this movie worse',
 'movie worse would',
 'worse would have',
 'would have beenoh',
 'have beenoh wait',
 'beenoh wait nono',
 'wait nono its',
 'nono its already',
 'its already as',
 'already as bad',
 'as bad as',
 'bad as it',
 'as it can',
 'it can get']
```

In [5]:
```
def one_hot_encode(tokens, num_words):
    token_index = {}
    for word in tokens:
        if word not in token_index:
            token_index[word] = len(token_index) + 1
    max_length = num_words

    results = np.zeros((max_length, max(token_index.values())+1))
    for i, word in enumerate(tokens[:max_length]):
        index = token_index.get(word)
        results[i, index] = 1.
    return results
one_hot_encode(y, 100)
```

Out[5]:
```
array([[0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

10.2

In [6]:
```
imdb_dir = '../../../data/external/imdb/aclImdb'
train_dir = os.path.join(imdb_dir, 'train')
labels = []
texts = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(train_dir, label_type)
    for fname in os.listdir(dir_name):
        if fname[-4:] == '.txt':
            with open(os.path.join(dir_name, fname)) as f:
                texts.append(f.read())
                if label_type == 'neg':
                    labels.append(0)
                else:
                    labels.append(1)
```

In [7]:
```
len(texts)
```

Out[7]:
```
25000
```

In [9]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

max_len = 250
training_samples = 10000
validation_samples = 10000
max_words = 10000

tokenizer = Tokenizer(num_words= 10000)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

```
Found 88582 unique tokens.
```

In [12]:
```
data = pad_sequences(sequences, maxlen= 250)
labels = np.asarray(labels)

data.shape,labels.shape
```

Out[12]:
```
((25000, 250), (25000,))
```

In [15]:
```
indicies = np.arange(data.shape[0])
np.random.shuffle(indicies)
data = data[indicies]
labels = labels[indicies]
```

In [16]:
```
x_train = data[:10000]
y_train = labels[:10000]
x_val = data[10000:20000]
y_val = labels[10000:20000]
```

In [18]:
```
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense

model = Sequential()
model.add(Embedding(10000, 100, input_length= 250))
model.add(Flatten())
model.add(Dense(32, activation= 'relu'))
model.add(Dense(1, activation= 'sigmoid'))
model.summary()
```

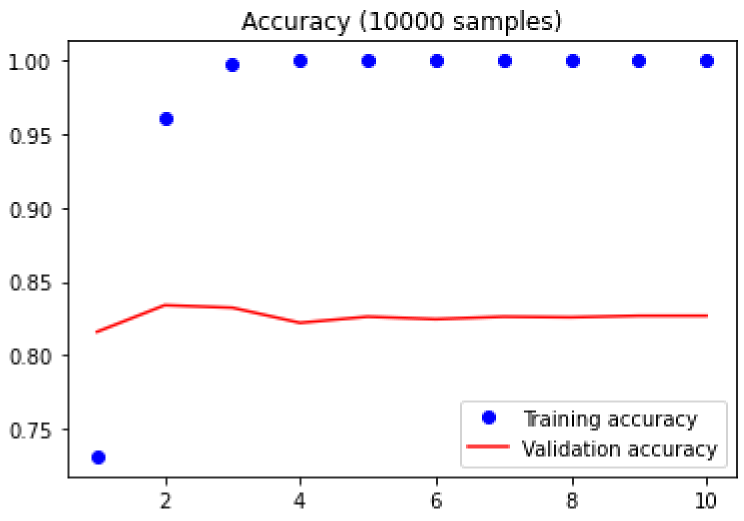
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 250, 100)          1000000   
_________________________________________________________________
flatten_1 (Flatten)          (None, 25000)             0         
_________________________________________________________________
dense_2 (Dense)              (None, 32)                800032    
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33        
=================================================================
Total params: 1,800,065
Trainable params: 1,800,065
Non-trainable params: 0
_________________________________________________________________
```

In [19]:
```
model.compile(optimizer= 'rmsprop',
             loss= 'binary_crossentropy',
             metrics= ['accuracy'])
history = model.fit(x_train, y_train, epochs= 10, batch_size= 32, validation_d
ata= (x_val, y_val))
```

```
Epoch 1/10
313/313 [==============================] - 4s 14ms/step - loss: 0.5029 - accu
racy: 0.7311 - val_loss: 0.4037 - val_accuracy: 0.8158
Epoch 2/10
313/313 [==============================] - 4s 11ms/step - loss: 0.1116 - accu
racy: 0.9610 - val_loss: 0.4017 - val_accuracy: 0.8339
Epoch 3/10
313/313 [==============================] - 4s 11ms/step - loss: 0.0102 - accu
racy: 0.9975 - val_loss: 0.5652 - val_accuracy: 0.8322
Epoch 4/10
313/313 [==============================] - 4s 12ms/step - loss: 6.1505e-04 -
accuracy: 0.9998 - val_loss: 0.7282 - val_accuracy: 0.8219
Epoch 5/10
313/313 [==============================] - 4s 12ms/step - loss: 1.9563e-05 -
accuracy: 1.0000 - val_loss: 0.8749 - val_accuracy: 0.8261
Epoch 6/10
313/313 [==============================] - 4s 11ms/step - loss: 1.3404e-07 -
accuracy: 1.0000 - val_loss: 0.9908 - val_accuracy: 0.8244
Epoch 7/10
313/313 [==============================] - 4s 12ms/step - loss: 1.4203e-08 -
accuracy: 1.0000 - val_loss: 1.0170 - val_accuracy: 0.8261
Epoch 8/10
313/313 [==============================] - 4s 12ms/step - loss: 5.6456e-09 -
accuracy: 1.0000 - val_loss: 1.0366 - val_accuracy: 0.8257
Epoch 9/10
313/313 [==============================] - 4s 12ms/step - loss: 4.0893e-09 -
accuracy: 1.0000 - val_loss: 1.0505 - val_accuracy: 0.8267
Epoch 10/10
313/313 [==============================] - 4s 12ms/step - loss: 3.1260e-09 -
accuracy: 1.0000 - val_loss: 1.0589 - val_accuracy: 0.8267
```

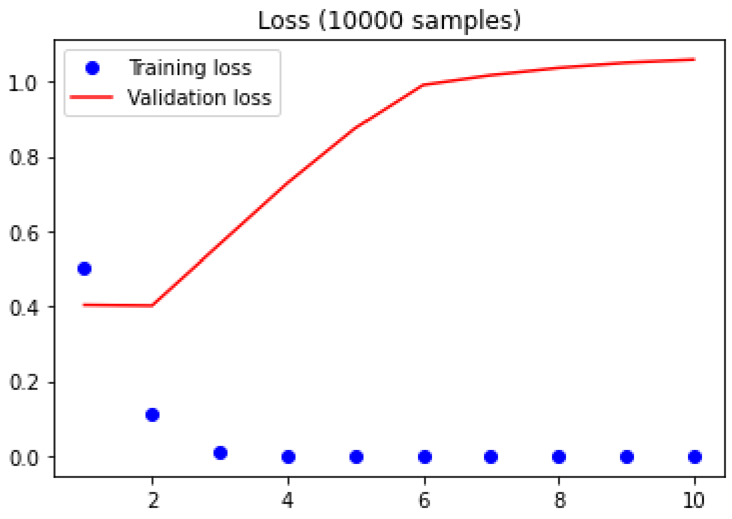
In [23]:
```
import matplotlib.pyplot as plt
epoch = range(1, len(history.history['accuracy']) +1)
plt.plot(epoch, history.history['accuracy'], 'bo', label= 'Training accuracy')
plt.plot(epoch, history.history['val_accuracy'], 'r', label= 'Validation accur
acy')
plt.legend()
plt.title('Accuracy (10000 samples)');
```

Out[23]:
```
Text(0.5, 1.0, 'Accuracy (10000 samples)')
```



In [24]:
```
plt.plot(epoch, history.history['loss'], 'bo', label= 'Training loss')
plt.plot(epoch, history.history['val_loss'], 'r', label= 'Validation loss')
plt.legend()
plt.title('Loss (10000 samples)');
```

Out[24]:
```
Text(0.5, 1.0, 'Loss (10000 samples)')
```



In [27]:
```
train_dir = os.path.join(imdb_dir, 'test')
test_labels = []
test_texts = []

for label_type in ['neg', 'pos']:
    dir_name = os.path.join(train_dir, label_type)
    for fname in os.listdir(dir_name):
        if fname[-4:] == '.txt':
            with open(os.path.join(dir_name, fname)) as f:
                test_texts.append(f.read())
                if label_type == 'neg':
                    test_labels.append(0)
                else:
                    test_labels.append(1)
test_sequences= tokenizer.texts_to_sequences(test_texts)
x_test= pad_sequences(test_sequences, maxlen= 250)
y_test= np.asarray(test_labels)
model.evaluate(x_test, y_test)
```

```
782/782 [==============================] - 2s 3ms/step - loss: 1.0114 - accur
acy: 0.8340
```

Out[27]:
```
[1.0113674402236938, 0.8339999914169312]
```

10.3

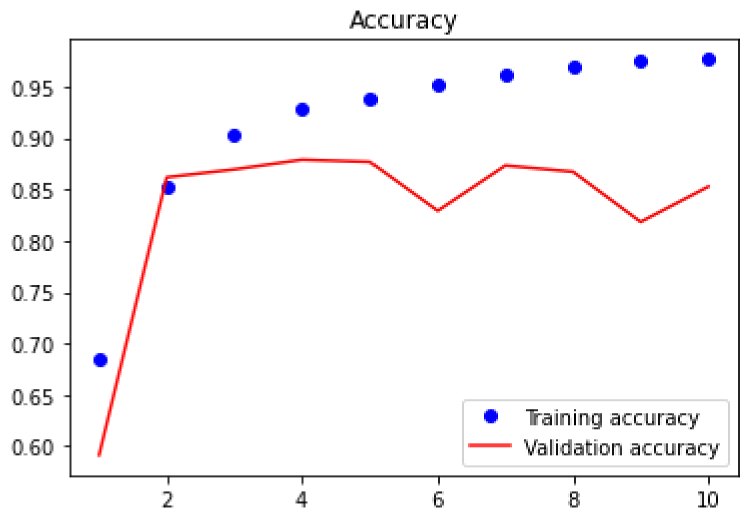
In [34]:
```
from keras.layers import LSTM
model = Sequential()
model.add(Embedding(10000, 32))
model.add(LSTM(32))
model.add(Dense(1, activation= 'sigmoid'))

model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_4 (Embedding)      (None, None, 32)          320000    
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                8320      
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 33        
=================================================================
Total params: 328,353
Trainable params: 328,353
Non-trainable params: 0
_________________________________________________________________
```

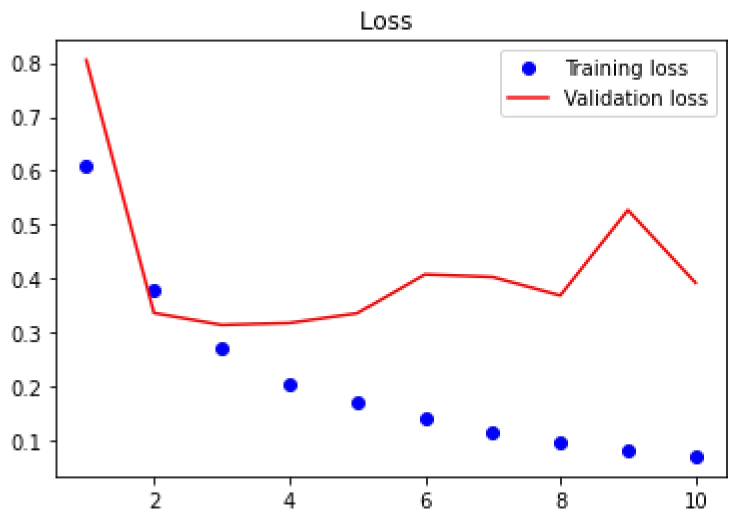
In [35]:
```
model.compile(optimizer= 'rmsprop',
             loss= 'binary_crossentropy',
             metrics= ['accuracy'])
history = model.fit(x_train, y_train, epochs= 10, batch_size= 128, validation_
data= (x_val,y_val))
```

```
Epoch 1/10
79/79 [==============================] - 22s 284ms/step - loss: 0.6071 - accu
racy: 0.6850 - val_loss: 0.8051 - val_accuracy: 0.5912
Epoch 2/10
79/79 [==============================] - 22s 275ms/step - loss: 0.3791 - accu
racy: 0.8529 - val_loss: 0.3358 - val_accuracy: 0.8618
Epoch 3/10
79/79 [==============================] - 22s 275ms/step - loss: 0.2691 - accu
racy: 0.9035 - val_loss: 0.3137 - val_accuracy: 0.8695
Epoch 4/10
79/79 [==============================] - 22s 276ms/step - loss: 0.2024 - accu
racy: 0.9280 - val_loss: 0.3173 - val_accuracy: 0.8789
Epoch 5/10
79/79 [==============================] - 22s 276ms/step - loss: 0.1690 - accu
racy: 0.9388 - val_loss: 0.3352 - val_accuracy: 0.8768
Epoch 6/10
79/79 [==============================] - 22s 273ms/step - loss: 0.1393 - accu
racy: 0.9518 - val_loss: 0.4075 - val_accuracy: 0.8292
Epoch 7/10
79/79 [==============================] - 21s 272ms/step - loss: 0.1129 - accu
racy: 0.9626 - val_loss: 0.4026 - val_accuracy: 0.8733
Epoch 8/10
79/79 [==============================] - 21s 268ms/step - loss: 0.0950 - accu
racy: 0.9698 - val_loss: 0.3684 - val_accuracy: 0.8672
Epoch 9/10
79/79 [==============================] - 21s 268ms/step - loss: 0.0796 - accu
racy: 0.9744 - val_loss: 0.5268 - val_accuracy: 0.8184
Epoch 10/10
79/79 [==============================] - 21s 268ms/step - loss: 0.0700 - accu
racy: 0.9768 - val_loss: 0.3920 - val_accuracy: 0.8527
```

In [36]:
```
epoch = range(1, len(history.history['accuracy']) +1)
plt.plot(epoch, history.history['accuracy'], 'bo', label= 'Training accuracy')
plt.plot(epoch, history.history['val_accuracy'], 'r', label= 'Validation accur
acy')
plt.legend()
plt.title('Accuracy');
```



In [37]:
```
plt.plot(epoch, history.history['loss'], 'bo', label= 'Training loss')
plt.plot(epoch, history.history['val_loss'], 'r', label= 'Validation loss')
plt.legend()
plt.title('Loss ');
```



10.4

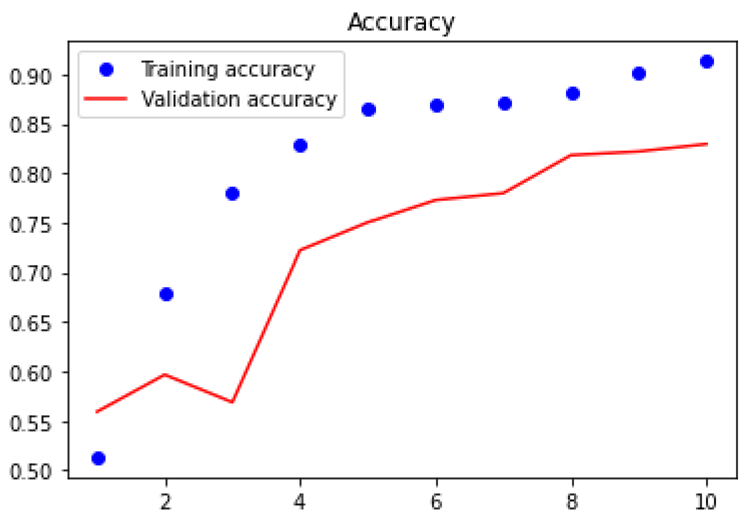
In [39]:
```python
from keras.optimizers import RMSprop
from keras import layers
model =Sequential()
model.add(Embedding(10000, 128, input_length= 250))
model.add(layers.Conv1D(32, 7, activation= 'relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation= 'relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(Dense(1))
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_6 (Embedding)      (None, 250, 128)          1280000
_________________________________________________________________
conv1d (Conv1D)              (None, 244, 32)           28704
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 48, 32)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 42, 32)            7200
_________________________________________________________________
global_max_pooling1d (Global (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,315,937
Trainable params: 1,315,937
Non-trainable params: 0
_________________________________________________________________
```

In [40]:
```
model.compile(optimizer= RMSprop(lr= 1e-4),
             loss= 'binary_crossentropy',
             metrics= ['accuracy'])
history = model.fit(x_train, y_train, epochs= 10, batch_size= 128, validation_
data= (x_val, y_val))
```

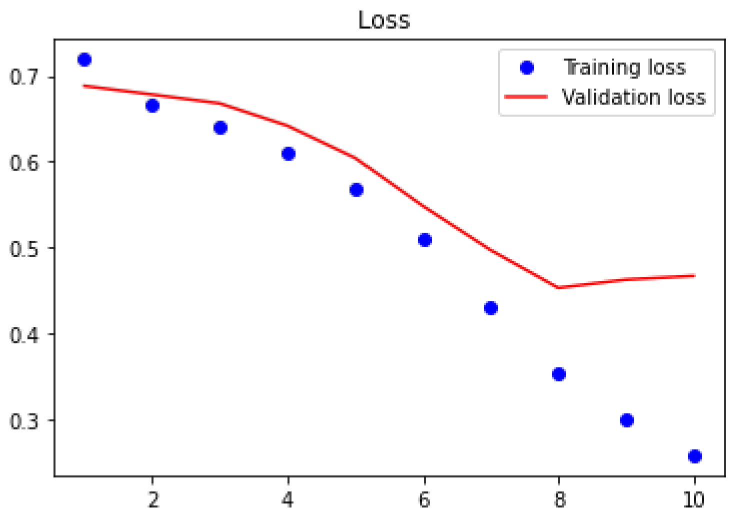
```
Epoch 1/10
79/79 [==============================] - 3s 43ms/step - loss: 0.7194 - accura
cy: 0.5133 - val_loss: 0.6882 - val_accuracy: 0.5594
Epoch 2/10
79/79 [==============================] - 3s 41ms/step - loss: 0.6674 - accura
cy: 0.6782 - val_loss: 0.6782 - val_accuracy: 0.5968
Epoch 3/10
79/79 [==============================] - 3s 41ms/step - loss: 0.6404 - accura
cy: 0.7805 - val_loss: 0.6679 - val_accuracy: 0.5689
Epoch 4/10
79/79 [==============================] - 3s 41ms/step - loss: 0.6096 - accura
cy: 0.8291 - val_loss: 0.6419 - val_accuracy: 0.7221
Epoch 5/10
79/79 [==============================] - 3s 40ms/step - loss: 0.5686 - accura
cy: 0.8662 - val_loss: 0.6043 - val_accuracy: 0.7503
Epoch 6/10
79/79 [==============================] - 3s 41ms/step - loss: 0.5099 - accura
cy: 0.8688 - val_loss: 0.5483 - val_accuracy: 0.7728
Epoch 7/10
79/79 [==============================] - 3s 41ms/step - loss: 0.4310 - accura
cy: 0.8723 - val_loss: 0.4970 - val_accuracy: 0.7798
Epoch 8/10
79/79 [==============================] - 3s 42ms/step - loss: 0.3545 - accura
cy: 0.8813 - val_loss: 0.4529 - val_accuracy: 0.8182
Epoch 9/10
79/79 [==============================] - 3s 41ms/step - loss: 0.2990 - accura
cy: 0.9009 - val_loss: 0.4625 - val_accuracy: 0.8219
Epoch 10/10
79/79 [==============================] - 3s 41ms/step - loss: 0.2578 - accura
cy: 0.9133 - val_loss: 0.4669 - val_accuracy: 0.8294
```

In [41]:
```
epoch = range(1, len(history.history['accuracy']) +1)
plt.plot(epoch, history.history['accuracy'], 'bo', label= 'Training accuracy')
plt.plot(epoch, history.history['val_accuracy'], 'r', label= 'Validation accur
acy')
plt.legend()
plt.title('Accuracy');
```



In [42]:
```
plt.plot(epoch, history.history['loss'], 'bo', label= 'Training loss')
plt.plot(epoch, history.history['val_loss'], 'r', label= 'Validation loss')
plt.legend()
plt.title('Loss ')
```

Out[42]:
```
Text(0.5, 1.0, 'Loss ')
```



In [ ]: