In [3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clie
nt_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.
com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapi
s.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.phot
os.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&
response_type=code

Enter your authorization code:
..........
Mounted at /content/drive
```

In [4]:
```
fname ='/content/drive/My Drive/asign-14-LSTM/'
!ls "/content/drive/My Drive/asign-14-LSTM"
```

```
vocab_cmn.pkl  word_idx.pkl  x_test.pkl  X_train.pkl  y_test.pkl  y_train.p
kl
```

In [0]:
```
# Credits: https://machinelearningmastery.com/sequence-classification-lstm-
recurrent-neural-networks-python-keras/
# LSTM for sequence classification in the IMDB dataset
import numpy
import pickle
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dropout, BatchNormalization
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.optimizers import Adam
from keras.regularizers import l2
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
```

[1] Convert AMZNFFReview data to IMDB format

In [0]:
```
 #%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np

import scipy
from collections import Counter
from itertools import islice
from sklearn.model_selection import train_test_split
from bs4 import BeautifulSoup
import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
import nltk
from nltk import word_tokenize, sent_tokenize

import re
import pickle

from tqdm import tqdm
import os
import time

class asign_14_lstm:

        def __init__(self):
                self.X_train = []
                self.x_test = []
                self.X_trn = pd.DataFrame()  # temporary
                self.x_tst = pd.DataFrame()  # temporary
                self.y_train = []
                self.y_test = []
                self.vocab_cmn = []
                self.vocab_all = []
                self.top_wrd_cnt = None
                self.wrd_idx = dict()
                self.X_trn_lst = []

        @property
        def X_train(self):
            return self._X_train

        @X_train.setter
        def X_train(self,new_X_train):
            self._X_train = new_X_train

        @property
        def X_trn(self):
            return self._X_trn

        @X_trn.setter
        def X_trn(self,new_X_trn):
```

```
            self._X_trn = new_X_trn

        @property
        def X_tst(self):
            return self._X_tst

        @X_tst.setter
        def X_trn(self,new_X_tst):
            self._X_tst = new_X_tst

        # Give reviews with Score>3 a positive rating(1), and reviews with
 a score<3 a negative rating(0).
        def partition(self,x):
                if x < 3:
                        return 0
                return 1

        def write_data(self,fnme,opdata):

                #fname = 'E:/appliedaicourse/assignments/dblite/kdtree_50
k/' + fnme
                fname = 'E:/appliedaiacourse/assignments/asign-14-LSTM/' +
fnme
                with open(fname, 'wb') as fp:
                        pickle.dump(opdata, fp)

        def decontracted(self,phrase):
                # specific
                phrase = re.sub(r"won't", "will not", phrase)
                phrase = re.sub(r"can\'t", "can not", phrase)

                # general
                phrase = re.sub(r"n\'t", " not", phrase)
                phrase = re.sub(r"\'re", " are", phrase)
                phrase = re.sub(r"\'s", " is", phrase)
                phrase = re.sub(r"\'d", " would", phrase)
                phrase = re.sub(r"\'ll", " will", phrase)
                phrase = re.sub(r"\'t", " not", phrase)
                phrase = re.sub(r"\'ve", " have", phrase)
                phrase = re.sub(r"\'m", " am", phrase)

                return phrase



        # Combining all the above statements
        def rw_preproc(self,xdata):

                preprocessed_reviews = []

                # tqdm is for printing the status bar

                for sentance in tqdm(xdata.values):
                        sentance = re.sub(r"http\S+", "", sentance)
                        sentance = BeautifulSoup(sentance, 'lxml').get_text
()
                        sentance = self.decontracted(sentance)
```

```
                        sentance = re.sub("\S*\d\S*", "", sentance).strip()
                        sentance = re.sub('[^A-Za-z]+', ' ', sentance)
                        # https://gist.github.com/sebleier/554280
                        sentance = ' '.join(e.lower() for e in sentance.spl
it() )
                        preprocessed_reviews.append(sentance.strip())
                return preprocessed_reviews


        def getreviews(self, nrows):
                X_trn = pd.DataFrame()
                # using SQLite Table to read data.
                filepath = os.path.abspath('E:/appliedaiacourse/assignment
s/dblite/database.sqlite')
                assert os.path.exists(filepath), 'the file does not exist'
                con = sqlite3.connect(filepath)
                #filtered_data = pd.read_sql_query(""" SELECT * FROM Review
s WHERE Score != 3 LIMIT 50000""", con)
                if nrows == -1 :
                        # fetch all rows
                        filtered_data = pd.read_sql_query(""" SELECT * FROM
Reviews WHERE Score != 3 """ , con)
                else:

                        filtered_data = pd.read_sql_query(""" SELECT * FROM
Reviews WHERE Score != 3 LIMIT """ + str(nrows), con)

                #changing reviews with score less than 3 to be positive and
vice-versa
                actualScore = filtered_data['Score']
                positiveNegative = actualScore.map(self.partition)
                filtered_data['Score'] = positiveNegative
                #Sorting data according to ProductId in ascending order
                sorted_data=filtered_data.sort_values('Time', axis=0, ascen
ding=True, inplace=False, kind='quicksort', na_position='last')
                #Deduplication of entries
                final=sorted_data.drop_duplicates(subset={"UserId","Profile
Name","Time","Text"}, keep='first', inplace=False)
                final.shape

                final=final[final.HelpfulnessNumerator<=final.HelpfulnessDe
nominator]

                #Before starting the next phase of preprocessing lets see t
he number of entries left
                print(final.shape)

                #How many positive and negative reviews are present in our
 dataset?
                final['Score'].value_counts()

                X_trn, X_tst, self.y_train, self.y_test = train_test_split(
final['Text'], final['Score'], stratify= final['Score'],test_size=0.2, rand
om_state=42)

                self.X_trn = self.rw_preproc(X_trn)
                self.x_tst  = self.rw_preproc(X_tst)
```

```
        def crea_vocab(self):
                k =0
                for sent in self.X_trn:
                        words =sent.split()
                        self.vocab_all += words
                        k += 1
                        print('crea_vocb fn processing ', k, 'th row')

        def crea_imdb_fmt(self,xdata, xlist):
                k=0
                for sent in xdata:
                        words = sent.split()
                        tmplst =[]
                        for  word  in words:
                                if word in self.wrd_idx:
                                        tmplst.append(self.wrd_idx[word])
                                else:
                                        tmplst.append(0)
                        k += 1
                        xlist.append(tmplst)
                        print('crea_imdb_fmt fn processing ', k, 'th row')
                return xlist



if __name__ == "__main__" :

        print('Process Starting')

        #instantiate the class
        ls_tm = asign_14_lstm()

        #get the reviews from db
        ls_tm.getreviews(50000)
        #max features
        ls_tm.top_wrd_cnt = 5000

        #create vocabulary
        ls_tm.crea_vocab()

        count= Counter(ls_tm.vocab_all)

        #get most common words
        ls_tm.vocab_cmn = count.most_common(ls_tm.top_wrd_cnt)

        i = 1

        for wrd, frq in ls_tm.vocab_cmn:
                ls_tm.wrd_idx[wrd] = i
                i += 1

        xlist =[]
        ls_tm.X_train = ls_tm.crea_imdb_fmt(ls_tm.X_trn,xlist)

        xlist =[]
```

```
        ls_tm.x_test = ls_tm.crea_imdb_fmt(ls_tm.x_tst,xlist)

        print(ls_tm.X_train[1])
        print(type(ls_tm.X_train[1]))
        print(len(ls_tm.X_train[1]))
        print(ls_tm.x_test[1])
        print(type(ls_tm.x_test[1]))
        print(len(ls_tm.x_test[1]))

        ls_tm.write_data('vocab_cmn.pkl',ls_tm.vocab_cmn)
        ls_tm.write_data('word_idx.pkl', ls_tm.wrd_idx)
        ls_tm.write_data('X_train.pkl',ls_tm.X_train)
        ls_tm.write_data('x_test.pkl',ls_tm.x_test)
        ls_tm.write_data('y_train.pkl',ls_tm.y_train)
        ls_tm.write_data('y_test.pkl',ls_tm.y_test)
```

[1.1] Loading the AMZNFFReview dataset

In [0]:
```
with open(fname+'X_train.pkl', 'rb') as fp:
    X_train = pickle.load(fp)

with open(fname+'x_test.pkl', 'rb') as fp1:
  x_test = pickle.load(fp1)

with open(fname+'y_train.pkl', 'rb') as fp2:
  y_train = pickle.load(fp2)

with open(fname+'y_test.pkl', 'rb') as fp3:
  y_test = pickle.load(fp3)
```

In [7]:
```
from keras import backend
backend.clear_session()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:107: The name tf.reset_default_graph is deprecated.
Please use tf.compat.v1.reset_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:111: The name tf.placeholder_with_default is deprec
ated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:66: The name tf.get_default_graph is deprecated. Pl
ease use tf.compat.v1.get_default_graph instead.
```

In [8]:

```
# truncate and/or pad input sequences
max_review_length = 600
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
x_test = sequence.pad_sequences(x_test, maxlen=max_review_length)

print(X_train.shape)
print(X_train[1])
```

```
(36856, 600)
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    9  327  100  177 2361    2  532
  425   15    9  327  683 3142   62 2618  683   19 3015    3  683   16
  687   98   30  541    3    2 1063  275    9   11    1 1416    3    5
  401    5    6  847    3  255  133   25 1066   18    2  197    5   36
   93    3   13  305 1277    5   67   37  375  252   33  348   12    4
  452    3    2   46  196   43   35  263   54    8    9   12    1  894
  248  129  208    9  327   36  430  607    3  583 4744  481]
```

In [0]:
```
%matplotlib notebook
import matplotlib.pyplot as plt
import numpy as np
import time
# https://gist.github.com/greydanus/f6eee59eaf1d90fcb3b534a25362cea4
# https://stackoverflow.com/a/14434334
# this function is used to update the plots for each epoch and error
def plt_mdl_res(x_val, trn_los, tst_los, tst_scr, tst_acc):
  # Visualize loss history
  plt.figure(figsize=(16,16))
  plt.plot(x_val, trn_los, 'r--')
  plt.plot(x_val, tst_los, 'b-')
  plt.title('Training Vs Test Loss')
  plt.legend(['Training Loss', 'Test Loss'])
  plt.xlabel('Epoch')
  plt.ylabel('Loss')
  plt.grid()
  plt.show();
  print('Test score:', tst_scr)
  print('Test accuracy:', tst_acc)

def plt_mdl_acc(x_val, trn_acc, tst_acc):
  # Visualize loss history
  plt.figure(figsize=(16,16))
  plt.plot(x_val, trn_acc, 'r--')
  plt.plot(x_val, tst_acc, 'b-')
  plt.title('Training Vs Test Accuracy')
  plt.legend(['Training Accuracy', 'Test Accuracy'])
  plt.xlabel('Epoch')
  plt.ylabel('Accuracy')
  plt.grid()
  plt.show();
```

[2] Single Level LSTM

[2.1]No Dropout with kernel_regularization and Batch_Normalization

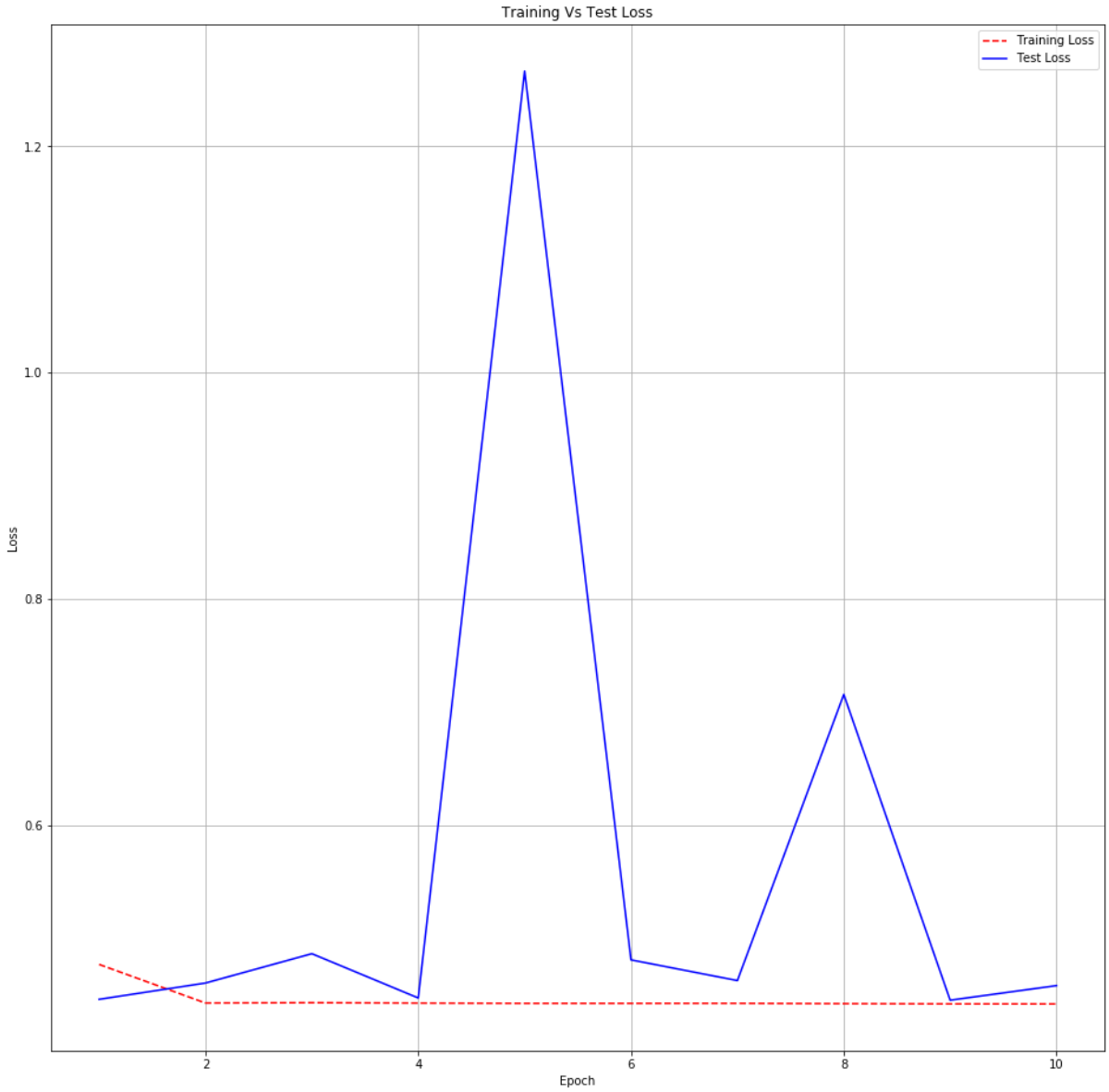
In [0]:
```
import keras
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 10
batch_size = 64
mdl_1 = Sequential()
optimize = keras.optimizers.Adam()
optimize.learning_rate=0.0000000001
mdl_1.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
#mdl_1.add(LSTM(100)) overfitting -- reducing number of units to 64
mdl_1.add(LSTM(16,kernel_regularizer=l2(0.000001)))   #overfitting -- reduc
ing number of units to 32
mdl_1.add(BatchNormalization())
#mdl_1.add(LSTM(16)) train and test accuracy are the same
mdl_1.add(Dense(1, activation='sigmoid'))
mdl_1.compile(loss='binary_crossentropy', optimizer=optimize, metrics=['acc
uracy'])
print(mdl_1.summary())
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_2 (LSTM)                (None, 16)                3136
_________________________________________________________________
batch_normalization_2 (Batch (None, 16)                64
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 17
=================================================================
Total params: 163,249
Trainable params: 163,217
Non-trainable params: 32
_________________________________________________________________
None
```

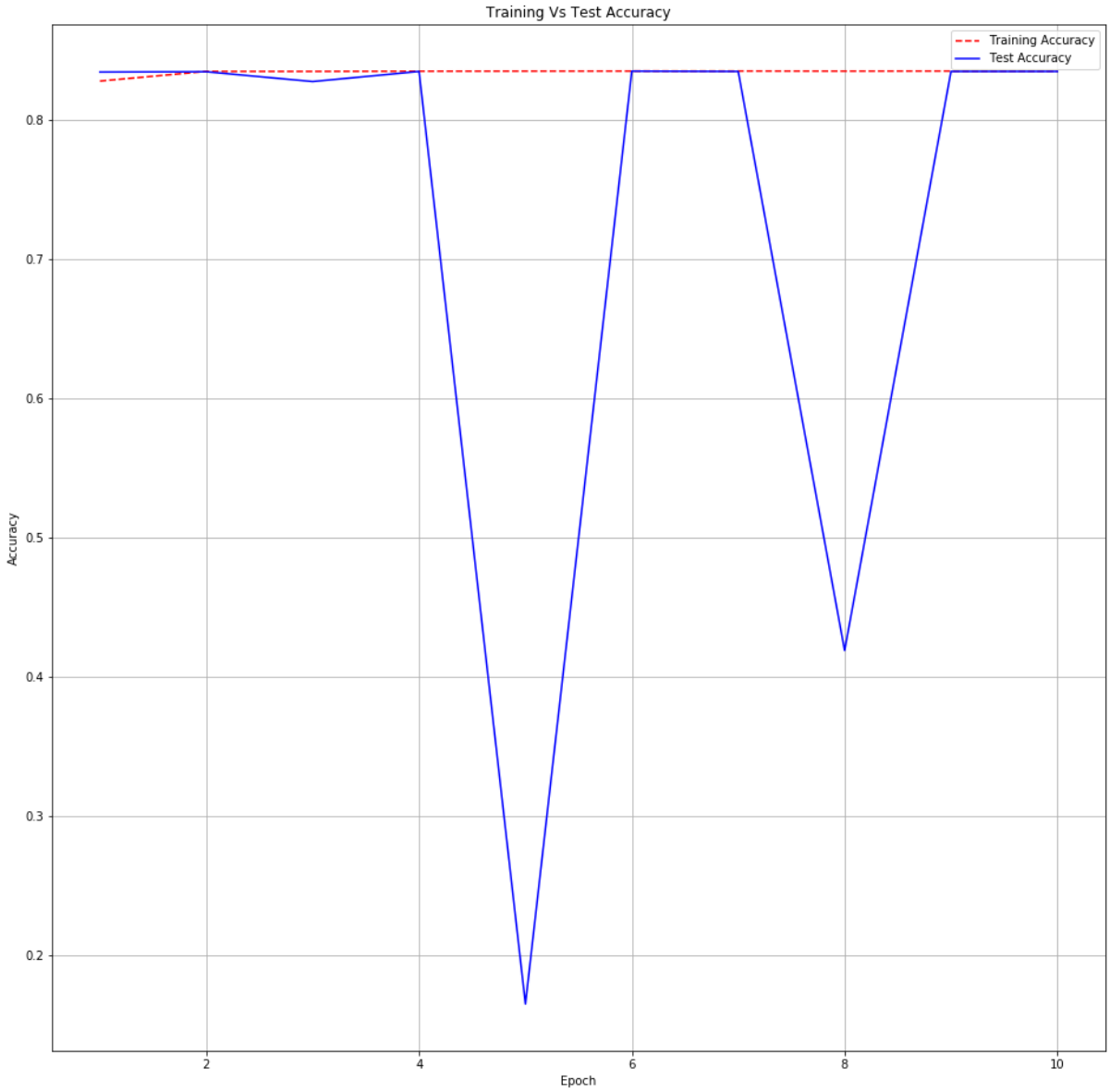
In [0]:
```
import keras
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logs_base_dir)
history_1 = mdl_1.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_size,verbose=1,validation_data=(x_test,y_test),callbacks=[tensorboard_callback] )
#history_1 = mdl_1.fit(X_train, y_train, nb_epoch=no_epoch, batch_size=batch_size,verbose=1,validation_data=(x_test,y_test) )
#callbacks=[keras.callbacks.TensorBoard(log_dir="logs/final/{}".format(time()), histogram_freq=1, write_graph=True, write_images=True)]
```

```
Train on 36856 samples, validate on 9215 samples
Epoch 1/10
36856/36856 [==============================] - 259s 7ms/step - loss: 0.4768 - acc: 0.8280 - val_loss: 0.4459 - val_acc: 0.8345
Epoch 2/10
36856/36856 [==============================] - 259s 7ms/step - loss: 0.4427 - acc: 0.8350 - val_loss: 0.4603 - val_acc: 0.8347
Epoch 3/10
36856/36856 [==============================] - 267s 7ms/step - loss: 0.4429 - acc: 0.8350 - val_loss: 0.4863 - val_acc: 0.8277
Epoch 4/10
36856/36856 [==============================] - 266s 7ms/step - loss: 0.4426 - acc: 0.8351 - val_loss: 0.4471 - val_acc: 0.8349
Epoch 5/10
36856/36856 [==============================] - 267s 7ms/step - loss: 0.4424 - acc: 0.8352 - val_loss: 1.2666 - val_acc: 0.1648
Epoch 6/10
36856/36856 [==============================] - 266s 7ms/step - loss: 0.4424 - acc: 0.8352 - val_loss: 0.4808 - val_acc: 0.8352
Epoch 7/10
36856/36856 [==============================] - 266s 7ms/step - loss: 0.4424 - acc: 0.8352 - val_loss: 0.4625 - val_acc: 0.8349
Epoch 8/10
36856/36856 [==============================] - 266s 7ms/step - loss: 0.4421 - acc: 0.8352 - val_loss: 0.7154 - val_acc: 0.4189
Epoch 9/10
36856/36856 [==============================] - 264s 7ms/step - loss: 0.4419 - acc: 0.8352 - val_loss: 0.4452 - val_acc: 0.8349
Epoch 10/10
36856/36856 [==============================] - 268s 7ms/step - loss: 0.4418 - acc: 0.8352 - val_loss: 0.4581 - val_acc: 0.8349
```

In [0]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt
no_epoch = 10
epoch_count = list(range(1,no_epoch+1))
# Get training and test loss histories
training_loss = history_1.history['loss']
test_loss = history_1.history['val_loss']
# Create count of the number of epochs
score = mdl_1.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

train_acc = history_1.history['acc']
test_acc = history_1.history['val_acc']
plt_mdl_acc(epoch_count,train_acc,test_acc)
```




```
Test score: 0.45805796802658394
Test accuracy: 0.8349430275881866
```



[2.1.2]More epochs

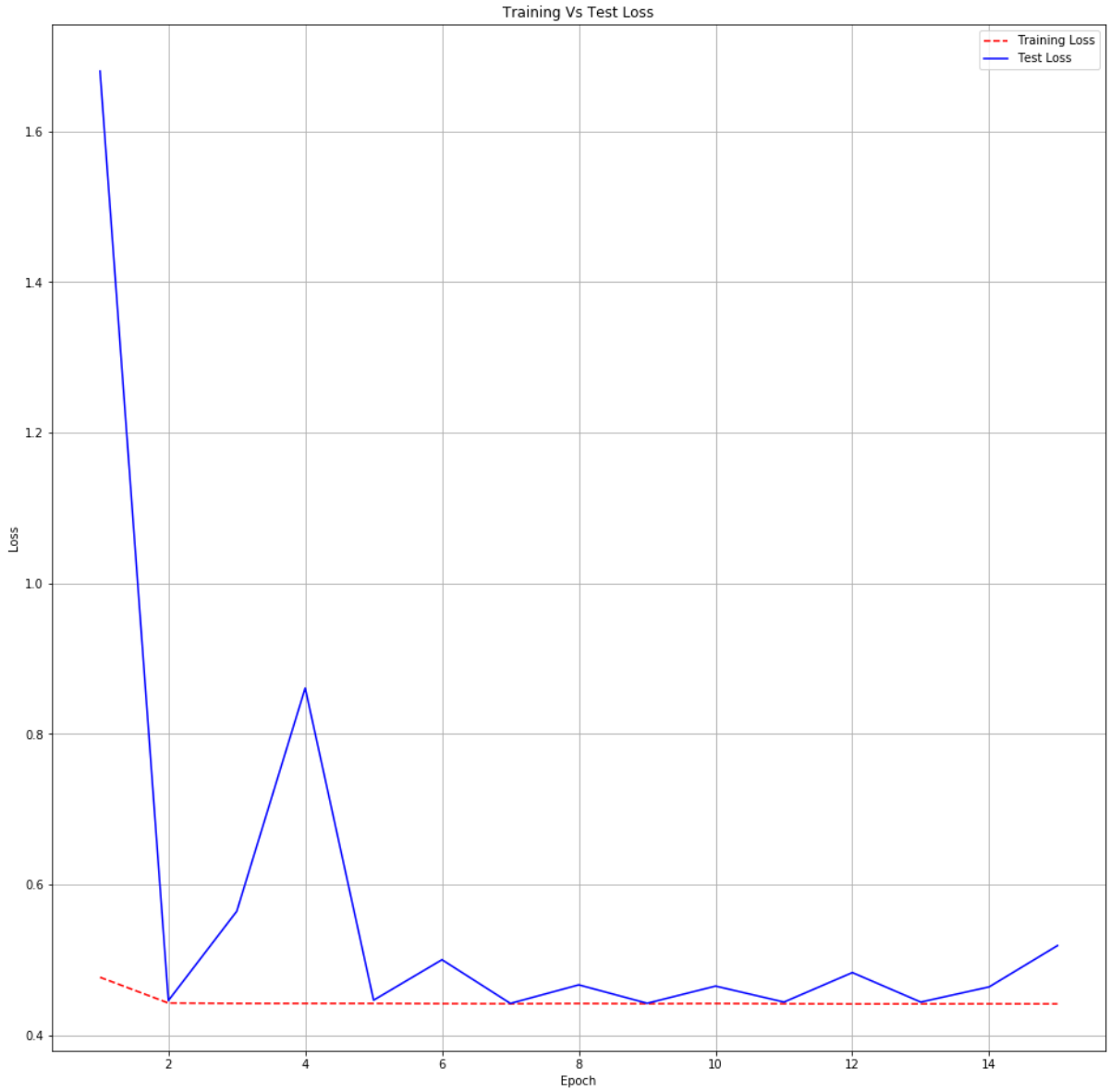
In [0]:
```
import keras
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 15
batch_size = 64
mdl_1 = Sequential()
optimize = keras.optimizers.Adam()
optimize.learning_rate=0.0000000001
mdl_1.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
#mdl_1.add(LSTM(100)) overfitting -- reducing number of units to 64
mdl_1.add(LSTM(16,kernel_regularizer=l2(0.000001)))   #overfitting -- reduc
ing number of units to 32
mdl_1.add(BatchNormalization())
#mdl_1.add(LSTM(16)) train and test accuracy are the same
mdl_1.add(Dense(1, activation='sigmoid'))
mdl_1.compile(loss='binary_crossentropy', optimizer=optimize, metrics=['acc
uracy'])
print(mdl_1.summary())
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_3 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_3 (LSTM)                (None, 16)                3136
_________________________________________________________________
batch_normalization_3 (Batch (None, 16)                64
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 163,249
Trainable params: 163,217
Non-trainable params: 32
_________________________________________________________________
None
```

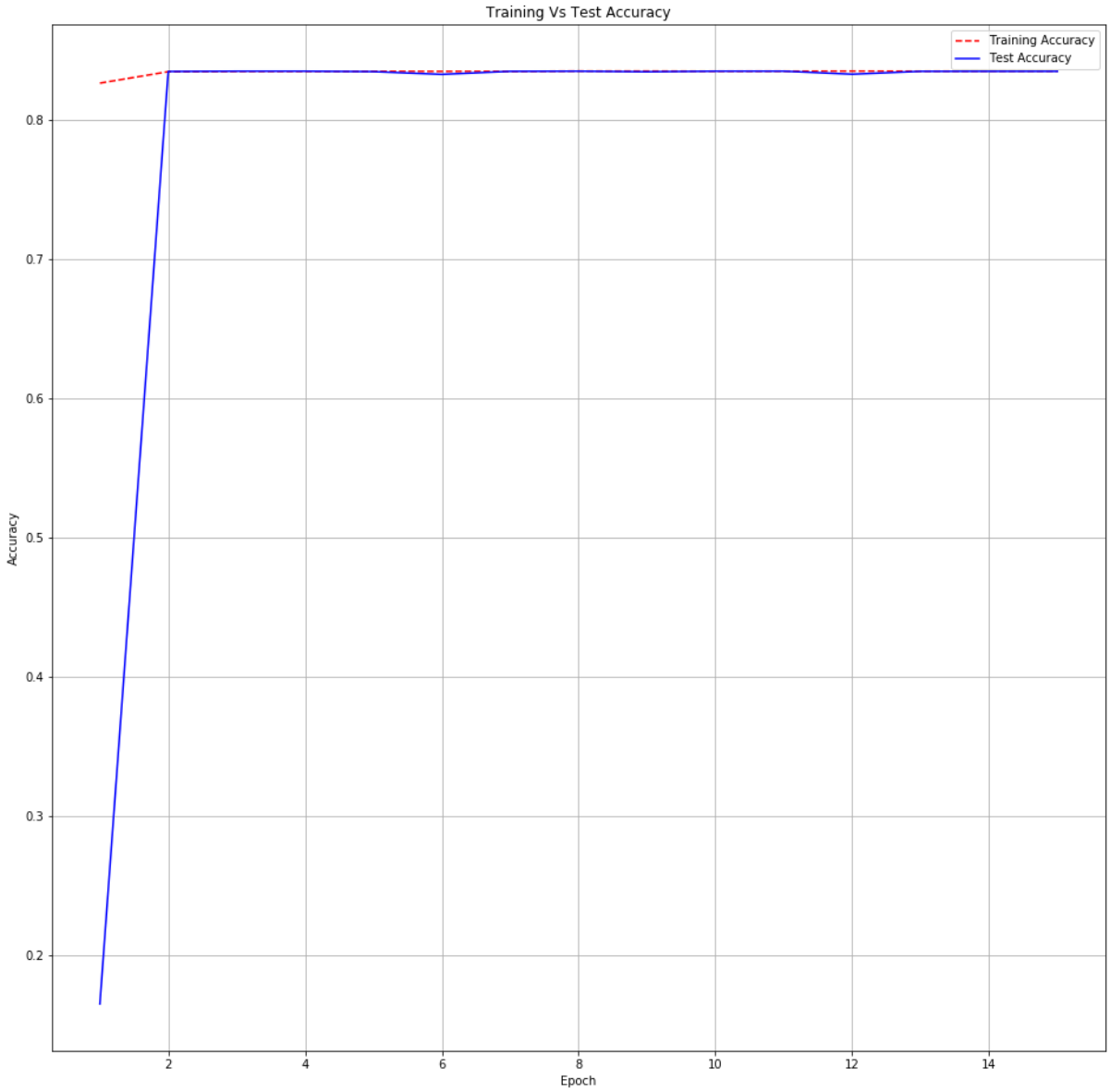
In [0]:
```
import keras
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logs_base_dir, h
istogram_freq=1, write_graph=True, write_images=True)
history_1 = mdl_1.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_s
ize,verbose=1,validation_data=(x_test,y_test),callbacks=[tensorboard_callba
ck] )
#history_1 = mdl_1.fit(X_train, y_train, nb_epoch=no_epoch, batch_size=batc
h_size,verbose=1,validation_data=(x_test,y_test) )
#callbacks=[keras.callbacks.TensorBoard(log_dir="logs/final/{}".format(time
()), histogram_freq=1, write_graph=True, write_images=True)]
```

```
Train on 36856 samples, validate on 9215 samples
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callba
cks.py:1068: The name tf.summary.histogram is deprecated. Please use tf.com
pat.v1.summary.histogram instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/callba
cks.py:1112: The name tf.summary.image is deprecated. Please use tf.compat.
v1.summary.image instead.

Epoch 1/15
36856/36856 [==============================] - 279s 8ms/step - loss: 0.4770
- acc: 0.8266 - val_loss: 1.6801 - val_acc: 0.1648
Epoch 2/15
36856/36856 [==============================] - 273s 7ms/step - loss: 0.4428
- acc: 0.8348 - val_loss: 0.4461 - val_acc: 0.8349
Epoch 3/15
36856/36856 [==============================] - 274s 7ms/step - loss: 0.4423
- acc: 0.8350 - val_loss: 0.5648 - val_acc: 0.8352
Epoch 4/15
36856/36856 [==============================] - 287s 8ms/step - loss: 0.4423
- acc: 0.8351 - val_loss: 0.8610 - val_acc: 0.8352
Epoch 5/15
36856/36856 [==============================] - 290s 8ms/step - loss: 0.4423
- acc: 0.8351 - val_loss: 0.4466 - val_acc: 0.8348
Epoch 6/15
36856/36856 [==============================] - 288s 8ms/step - loss: 0.4420
- acc: 0.8350 - val_loss: 0.5003 - val_acc: 0.8329
Epoch 7/15
36856/36856 [==============================] - 288s 8ms/step - loss: 0.4419
- acc: 0.8350 - val_loss: 0.4423 - val_acc: 0.8351
Epoch 8/15
36856/36856 [==============================] - 284s 8ms/step - loss: 0.4422
- acc: 0.8352 - val_loss: 0.4670 - val_acc: 0.8352
Epoch 9/15
36856/36856 [==============================] - 281s 8ms/step - loss: 0.4419
- acc: 0.8353 - val_loss: 0.4425 - val_acc: 0.8347
Epoch 10/15
36856/36856 [==============================] - 280s 8ms/step - loss: 0.4422
- acc: 0.8351 - val_loss: 0.4653 - val_acc: 0.8352
Epoch 11/15
36856/36856 [==============================] - 273s 7ms/step - loss: 0.4419
- acc: 0.8352 - val_loss: 0.4441 - val_acc: 0.8352
Epoch 12/15
36856/36856 [==============================] - 279s 8ms/step - loss: 0.4416
- acc: 0.8352 - val_loss: 0.4832 - val_acc: 0.8330
Epoch 13/15
36856/36856 [==============================] - 274s 7ms/step - loss: 0.4417
- acc: 0.8352 - val_loss: 0.4441 - val_acc: 0.8351
Epoch 14/15
36856/36856 [==============================] - 275s 7ms/step - loss: 0.4417
- acc: 0.8352 - val_loss: 0.4642 - val_acc: 0.8351
Epoch 15/15
36856/36856 [==============================] - 275s 7ms/step - loss: 0.4417
- acc: 0.8352 - val_loss: 0.5191 - val_acc: 0.8351
```

In [0]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt
no_epoch = 15
epoch_count = list(range(1,no_epoch+1))
# Get training and test loss histories
training_loss = history_1.history['loss']
test_loss = history_1.history['val_loss']
# Create count of the number of epochs
score = mdl_1.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

train_acc = history_1.history['acc']
test_acc = history_1.history['val_acc']
plt_mdl_acc(epoch_count,train_acc,test_acc)
```




```
Test score: 0.5191142415974493
Test accuracy: 0.8350515463076658
```



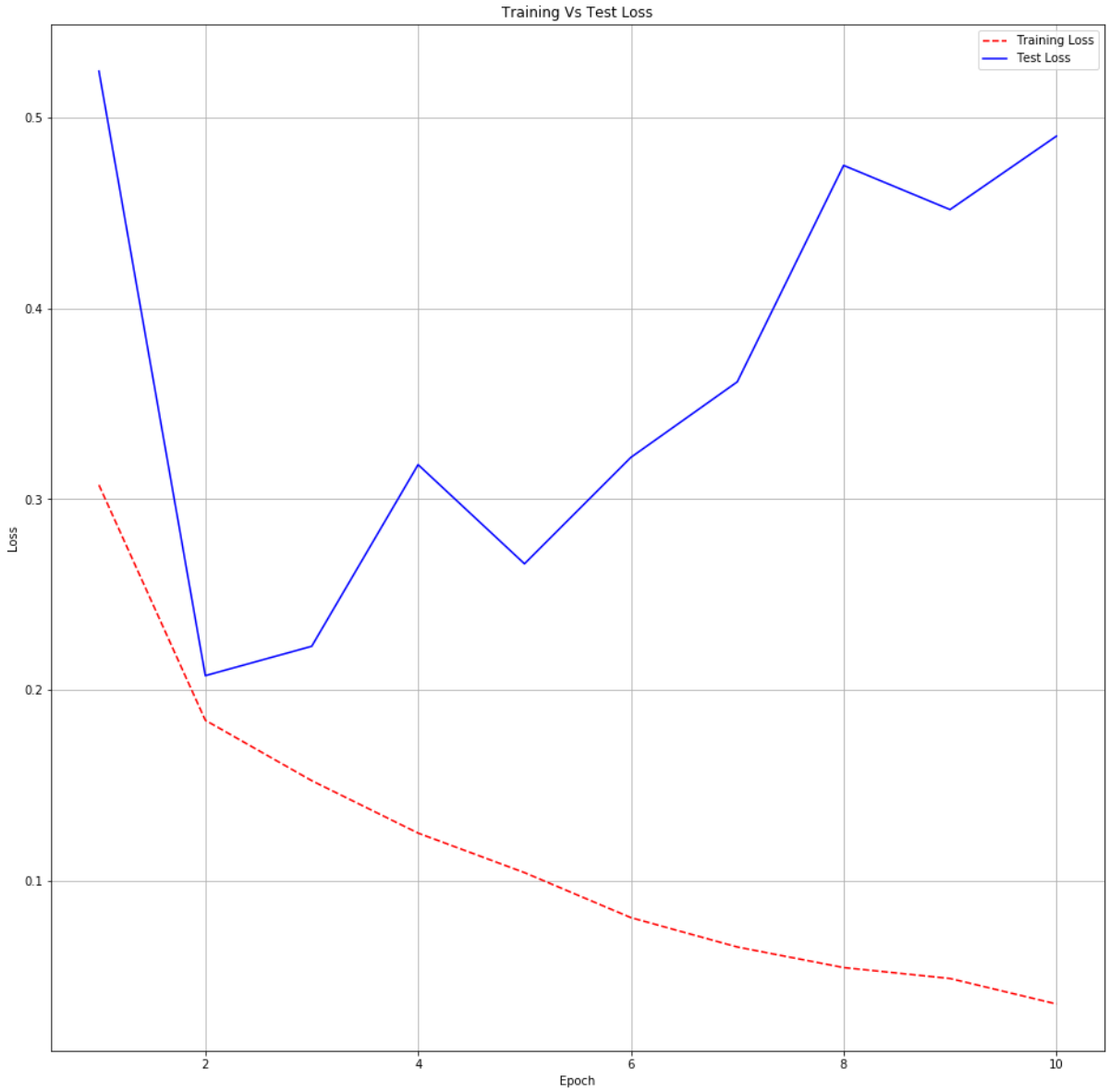
[2.1.3]More neurons

In [10]:
```
import keras
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 10
batch_size = 64
mdl_1 = Sequential()
optimize = keras.optimizers.Adam()
optimize.learning_rate=0.0000000001
mdl_1.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
#mdl_1.add(LSTM(100)) overfitting -- reducing number of units to 64
mdl_1.add(LSTM(32,kernel_regularizer=l2(0.000001)))   #overfitting -- reduc
ing number of units to 32
mdl_1.add(BatchNormalization())
#mdl_1.add(LSTM(16)) train and test accuracy are the same
mdl_1.add(Dense(1, activation='sigmoid'))
mdl_1.compile(loss='binary_crossentropy', optimizer=optimize, metrics=['acc
uracy'])
print(mdl_1.summary())
history_1 = mdl_1.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_s
ize,verbose=1,validation_data=(x_test,y_test) )
```

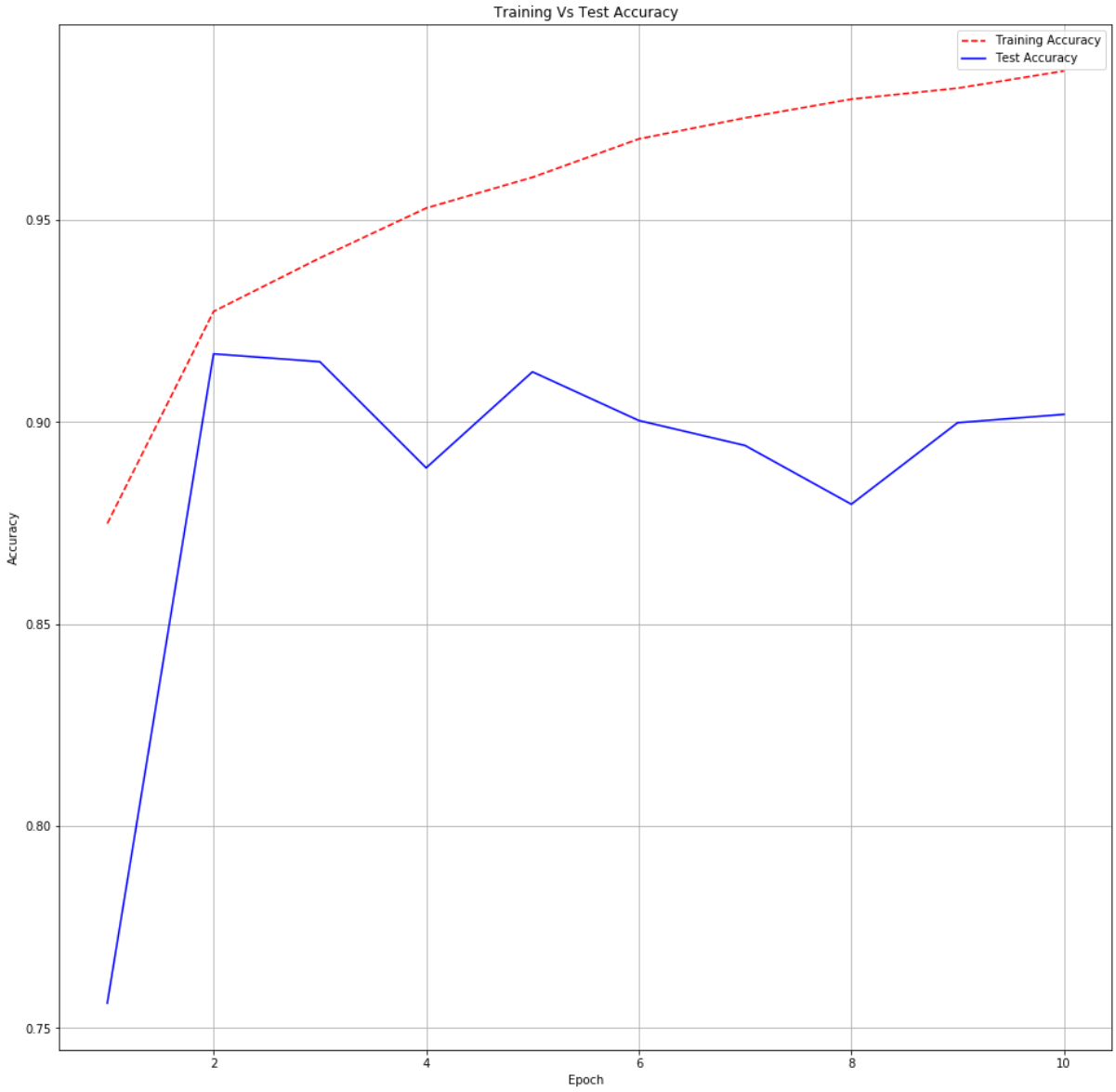
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:541: The name tf.placeholder is deprecated. Please
use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:4432: The name tf.random_uniform is deprecated. Ple
ase use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimi
zers.py:793: The name tf.train.Optimizer is deprecated. Please use tf.compa
t.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:3657: The name tf.log is deprecated. Please use tf.
math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_c
ore/python/ops/nn_impl.py:183: where (from tensorflow.python.ops.array_ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_1 (LSTM)                (None, 32)                8320
_________________________________________________________________
batch_normalization_1 (Batch (None, 32)                128
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 168,513
Trainable params: 168,449
Non-trainable params: 64
_________________________________________________________________
None
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:1033: The name tf.assign_add is deprecated. Please
use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:1020: The name tf.assign is deprecated. Please use
tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:3005: The name tf.Session is deprecated. Please use
tf.compat.v1.Session instead.

Train on 36856 samples, validate on 9215 samples
Epoch 1/10
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:190: The name tf.get_default_session is deprecated.
Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
```

```
d/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated. Please
use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:207: The name tf.global_variables is deprecated. Pl
ease use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:216: The name tf.is_variable_initialized is depreca
ted. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:223: The name tf.variables_initializer is deprecate
d. Please use tf.compat.v1.variables_initializer instead.

36856/36856 [==============================] - 236s 6ms/step - loss: 0.3074
- acc: 0.8749 - val_loss: 0.5244 - val_acc: 0.7562
Epoch 2/10
36856/36856 [==============================] - 235s 6ms/step - loss: 0.1841
- acc: 0.9274 - val_loss: 0.2075 - val_acc: 0.9169
Epoch 3/10
36856/36856 [==============================] - 235s 6ms/step - loss: 0.1524
- acc: 0.9406 - val_loss: 0.2229 - val_acc: 0.9149
Epoch 4/10
36856/36856 [==============================] - 235s 6ms/step - loss: 0.1249
- acc: 0.9530 - val_loss: 0.3181 - val_acc: 0.8887
Epoch 5/10
36856/36856 [==============================] - 235s 6ms/step - loss: 0.1042
- acc: 0.9606 - val_loss: 0.2661 - val_acc: 0.9124
Epoch 6/10
36856/36856 [==============================] - 235s 6ms/step - loss: 0.0806
- acc: 0.9701 - val_loss: 0.3219 - val_acc: 0.9004
Epoch 7/10
36856/36856 [==============================] - 235s 6ms/step - loss: 0.0652
- acc: 0.9753 - val_loss: 0.3615 - val_acc: 0.8942
Epoch 8/10
36856/36856 [==============================] - 232s 6ms/step - loss: 0.0544
- acc: 0.9799 - val_loss: 0.4749 - val_acc: 0.8797
Epoch 9/10
36856/36856 [==============================] - 233s 6ms/step - loss: 0.0487
- acc: 0.9826 - val_loss: 0.4518 - val_acc: 0.8998
Epoch 10/10
36856/36856 [==============================] - 234s 6ms/step - loss: 0.0354
- acc: 0.9869 - val_loss: 0.4903 - val_acc: 0.9019
```

In [12]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt
no_epoch = 10
epoch_count = list(range(1,no_epoch+1))
# Get training and test loss histories
training_loss = history_1.history['loss']
test_loss = history_1.history['val_loss']
# Create count of the number of epochs
score = mdl_1.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

train_acc = history_1.history['acc']
test_acc = history_1.history['val_acc']
plt_mdl_acc(epoch_count,train_acc,test_acc)
```




```
Test score: 0.49025379213233133
Test accuracy: 0.9018990776038208
```

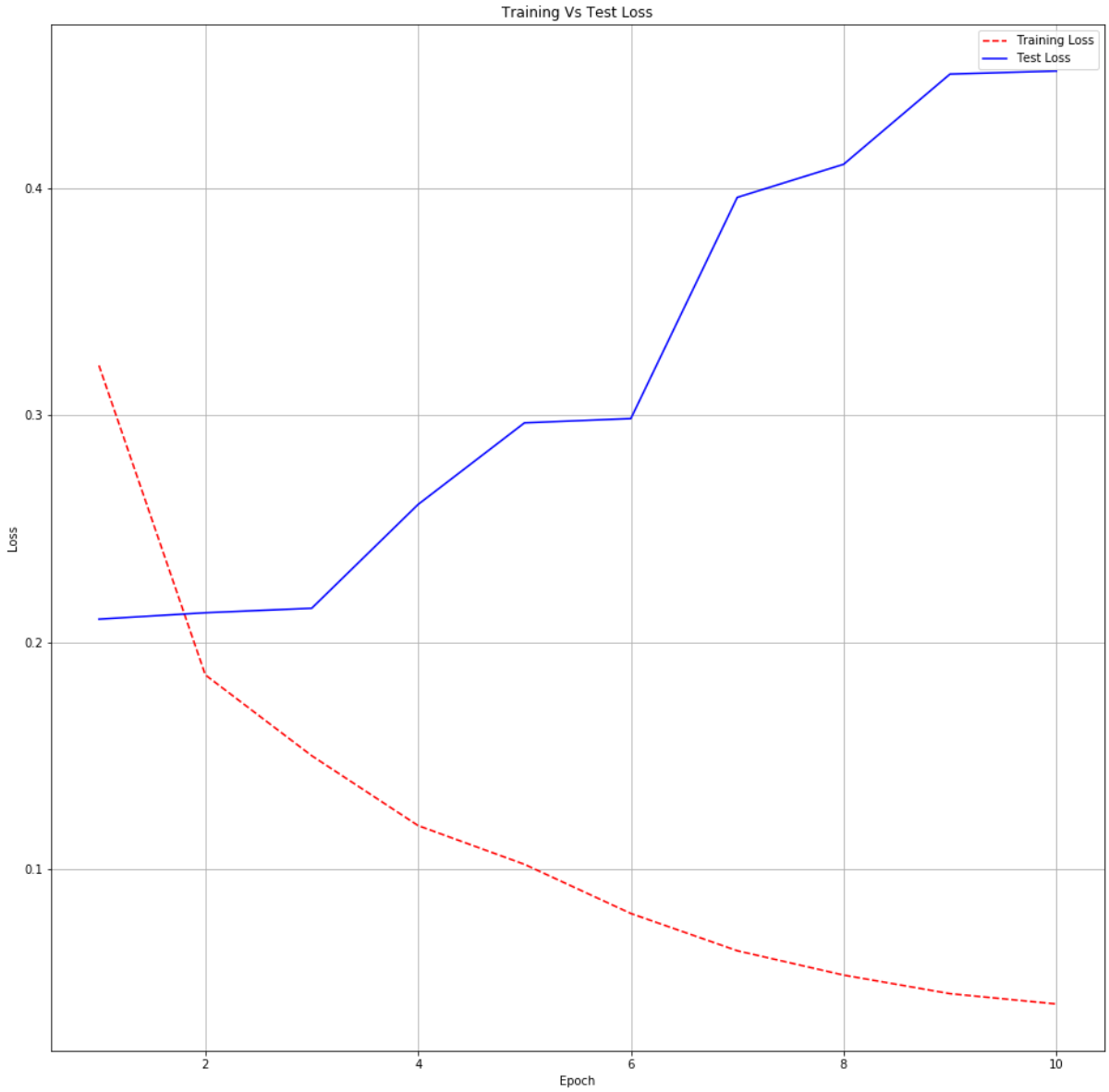


[2.1.4]More neurons with dropout

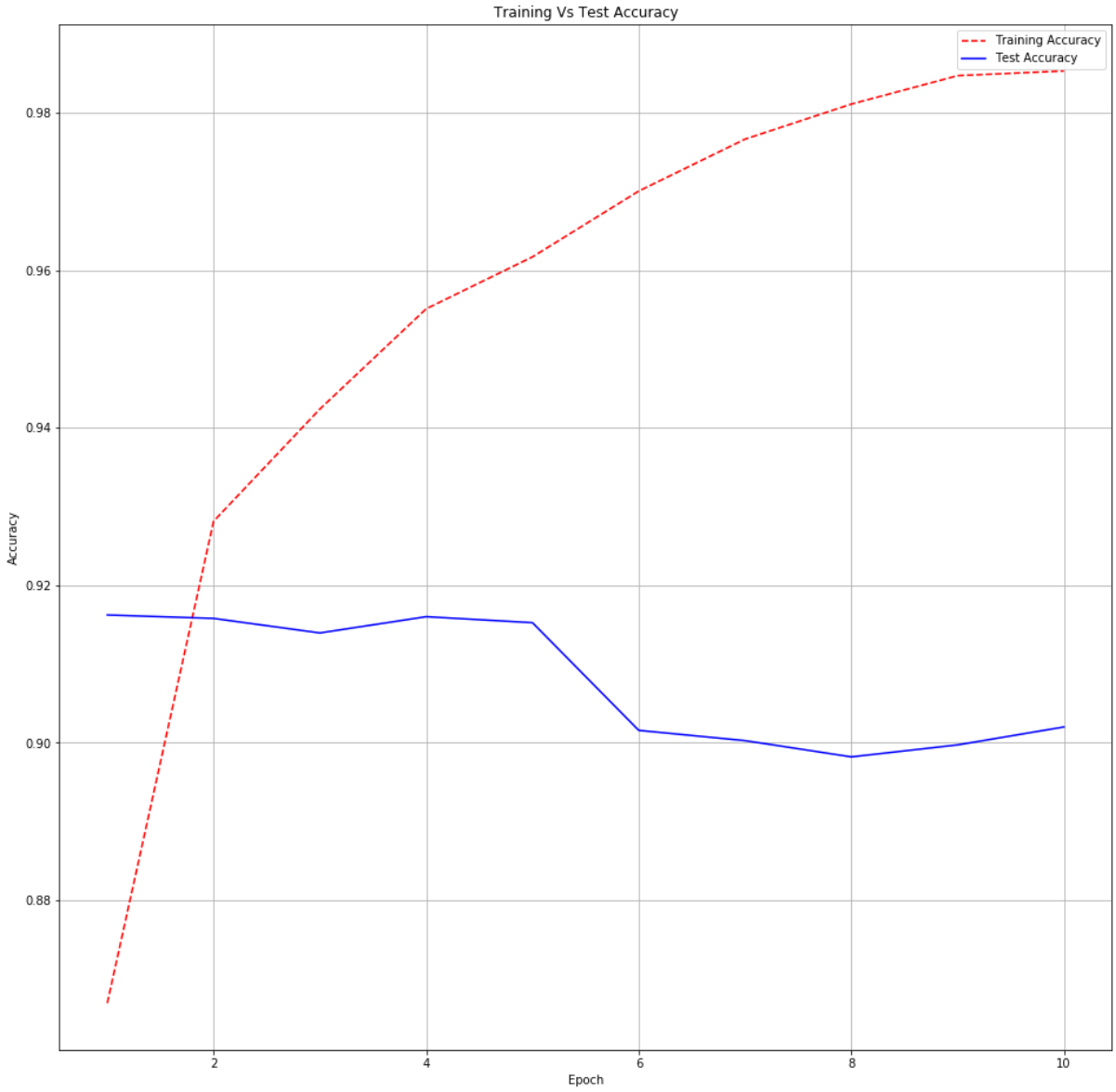
In [13]:
```
import keras
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 10
batch_size = 64
mdl_1 = Sequential()
optimize = keras.optimizers.Adam()
optimize.learning_rate=0.0000000001
mdl_1.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
#mdl_1.add(LSTM(100)) overfitting -- reducing number of units to 64
mdl_1.add(LSTM(32,kernel_regularizer=l2(0.000001)))   #overfitting -- reduc
ing number of units to 32
mdl_1.add(Dropout(rate=0.2))
mdl_1.add(BatchNormalization())
#mdl_1.add(LSTM(16)) train and test accuracy are the same
mdl_1.add(Dense(1, activation='sigmoid'))
mdl_1.compile(loss='binary_crossentropy', optimizer=optimize, metrics=['acc
uracy'])
print(mdl_1.summary())
history_1 = mdl_1.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_s
ize,verbose=1,validation_data=(x_test,y_test) )
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backen
d/tensorflow_backend.py:3733: calling dropout (from tensorflow.python.ops.n
n_ops) with keep_prob is deprecated and will be removed in a future versio
n.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_2 (LSTM)                (None, 32)                8320
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
batch_normalization_2 (Batch (None, 32)                128
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 168,513
Trainable params: 168,449
Non-trainable params: 64
_________________________________________________________________
None
Train on 36856 samples, validate on 9215 samples
Epoch 1/10
36856/36856 [==============================] - 238s 6ms/step - loss: 0.3219
- acc: 0.8669 - val_loss: 0.2101 - val_acc: 0.9162
Epoch 2/10
36856/36856 [==============================] - 242s 7ms/step - loss: 0.1856
- acc: 0.9281 - val_loss: 0.2130 - val_acc: 0.9158
Epoch 3/10
36856/36856 [==============================] - 249s 7ms/step - loss: 0.1500
- acc: 0.9424 - val_loss: 0.2149 - val_acc: 0.9139
Epoch 4/10
36856/36856 [==============================] - 252s 7ms/step - loss: 0.1192
- acc: 0.9551 - val_loss: 0.2607 - val_acc: 0.9160
Epoch 5/10
36856/36856 [==============================] - 238s 6ms/step - loss: 0.1022
- acc: 0.9617 - val_loss: 0.2966 - val_acc: 0.9152
Epoch 6/10
36856/36856 [==============================] - 243s 7ms/step - loss: 0.0805
- acc: 0.9700 - val_loss: 0.2985 - val_acc: 0.9016
Epoch 7/10
36856/36856 [==============================] - 242s 7ms/step - loss: 0.0641
- acc: 0.9766 - val_loss: 0.3959 - val_acc: 0.9003
Epoch 8/10
36856/36856 [==============================] - 241s 7ms/step - loss: 0.0534
- acc: 0.9811 - val_loss: 0.4105 - val_acc: 0.8982
Epoch 9/10
36856/36856 [==============================] - 242s 7ms/step - loss: 0.0452
- acc: 0.9847 - val_loss: 0.4502 - val_acc: 0.8997
Epoch 10/10
36856/36856 [==============================] - 242s 7ms/step - loss: 0.0407
- acc: 0.9853 - val_loss: 0.4515 - val_acc: 0.9020
```

In [14]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt
no_epoch = 10
epoch_count = list(range(1,no_epoch+1))
# Get training and test loss histories
training_loss = history_1.history['loss']
test_loss = history_1.history['val_loss']
# Create count of the number of epochs
score = mdl_1.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

train_acc = history_1.history['acc']
test_acc = history_1.history['val_acc']
plt_mdl_acc(epoch_count,train_acc,test_acc)
```




```
Test score: 0.45151769720984103
Test accuracy: 0.9020075963233
```


Training Vs Test Accuracy
Training Accuracy
Test Accuracy
Accuracy
Epoch
0.98
0.96
0.94
0.92
0.90
0.88
2
4
6
8
10

[2.1.5]Simple

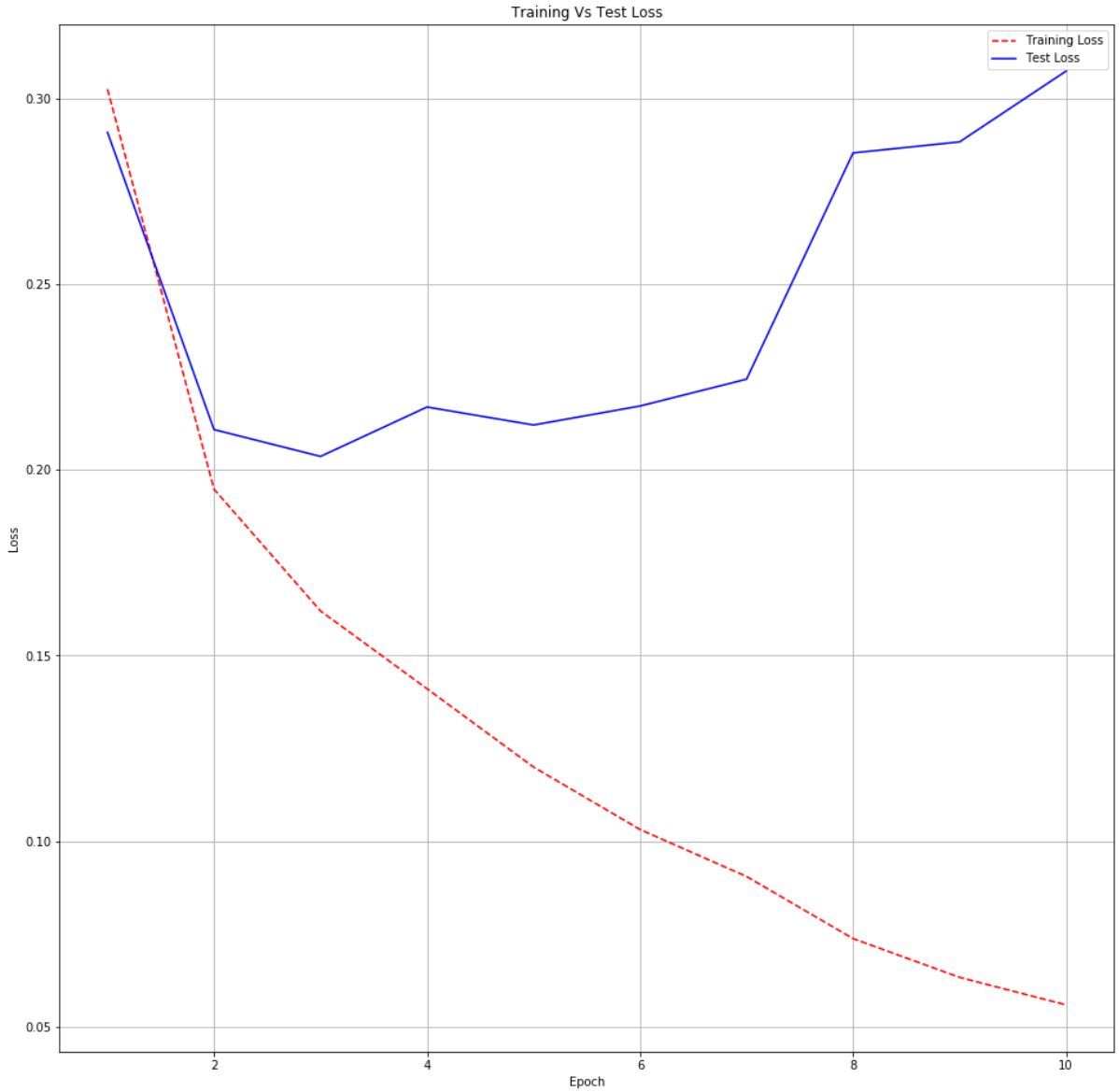
In [0]:
```
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 10
batch_size = 64
mdl_1 = Sequential()
mdl_1.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
mdl_1.add(LSTM(64))
mdl_1.add(Dense(1, activation='sigmoid'))
mdl_1.compile(loss='binary_crossentropy', optimizer=optimize, metrics=['acc
uracy'])
print(mdl_1.summary())
```

```
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_13 (Embedding)     (None, 600, 32)           160032
_________________________________________________________________
lstm_13 (LSTM)               (None, 64)                24832
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 65
=================================================================
Total params: 184,929
Trainable params: 184,929
Non-trainable params: 0
_________________________________________________________________
None
```

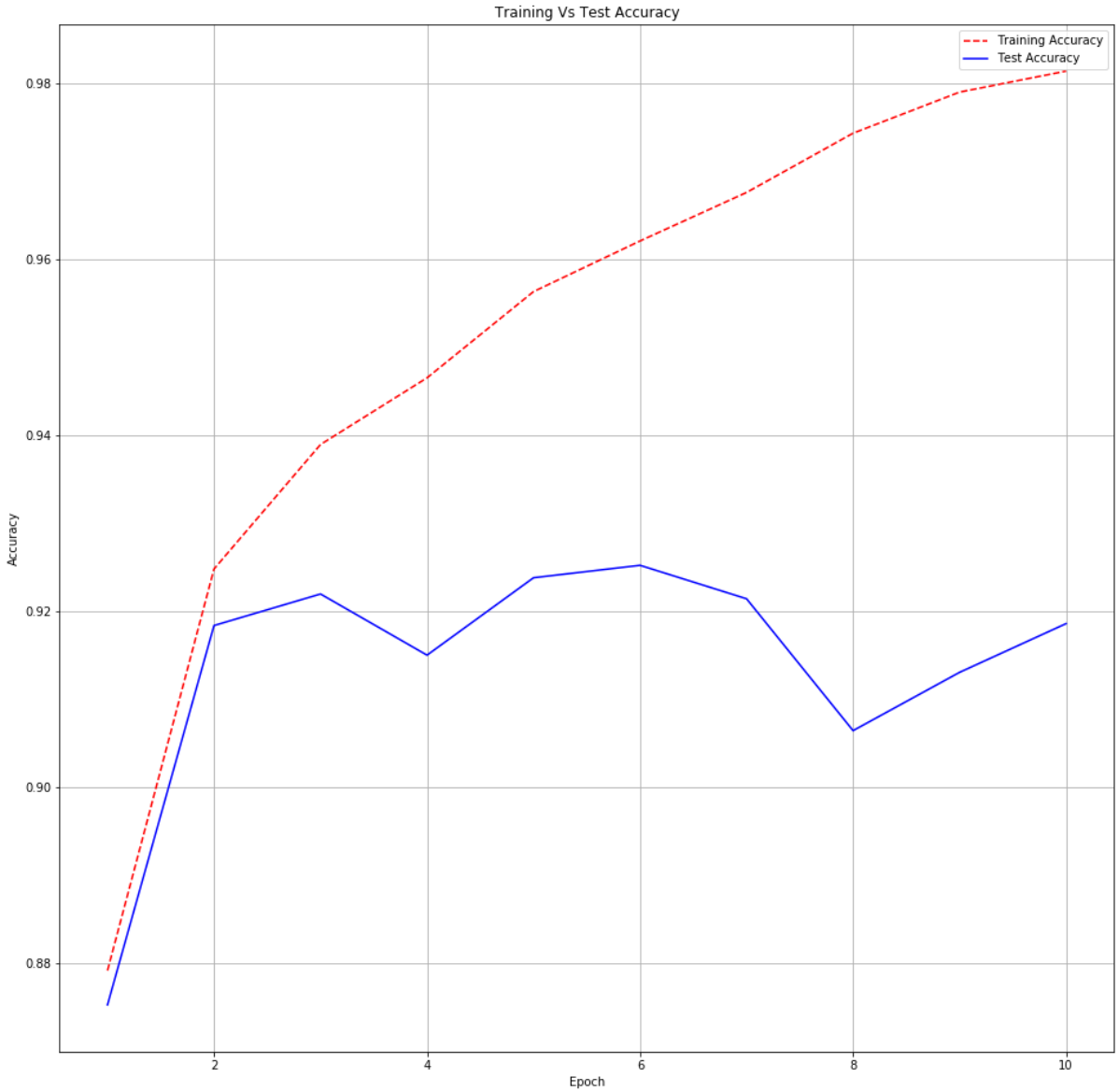
In [0]:
```
import keras
tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logs_base_dir)
history_1 = mdl_1.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_s
ize,verbose=1,validation_data=(x_test,y_test),callbacks=[tensorboard_callba
ck] )
#history_1 = mdl_1.fit(X_train, y_train, nb_epoch=no_epoch, batch_size=batc
h_size,verbose=1,validation_data=(x_test,y_test) )
```

```
Train on 36856 samples, validate on 9215 samples
Epoch 1/10
36856/36856 [==============================] - 373s 10ms/step - loss: 0.302
5 - acc: 0.8792 - val_loss: 0.2909 - val_acc: 0.8753
Epoch 2/10
36856/36856 [==============================] - 372s 10ms/step - loss: 0.194
8 - acc: 0.9248 - val_loss: 0.2108 - val_acc: 0.9184
Epoch 3/10
36856/36856 [==============================] - 369s 10ms/step - loss: 0.162
0 - acc: 0.9390 - val_loss: 0.2036 - val_acc: 0.9220
Epoch 4/10
36856/36856 [==============================] - 371s 10ms/step - loss: 0.141
0 - acc: 0.9465 - val_loss: 0.2169 - val_acc: 0.9150
Epoch 5/10
36856/36856 [==============================] - 370s 10ms/step - loss: 0.119
9 - acc: 0.9563 - val_loss: 0.2121 - val_acc: 0.9238
Epoch 6/10
36856/36856 [==============================] - 373s 10ms/step - loss: 0.103
1 - acc: 0.9621 - val_loss: 0.2172 - val_acc: 0.9252
Epoch 7/10
36856/36856 [==============================] - 373s 10ms/step - loss: 0.090
5 - acc: 0.9676 - val_loss: 0.2244 - val_acc: 0.9214
Epoch 8/10
36856/36856 [==============================] - 375s 10ms/step - loss: 0.073
7 - acc: 0.9743 - val_loss: 0.2853 - val_acc: 0.9065
Epoch 9/10
36856/36856 [==============================] - 372s 10ms/step - loss: 0.063
3 - acc: 0.9790 - val_loss: 0.2883 - val_acc: 0.9131
Epoch 10/10
36856/36856 [==============================] - 374s 10ms/step - loss: 0.055
9 - acc: 0.9814 - val_loss: 0.3074 - val_acc: 0.9186
```

In [0]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt
no_epoch = 10
epoch_count = list(range(1,no_epoch+1))
# Get training and test loss histories
training_loss = history_1.history['loss']
test_loss = history_1.history['val_loss']
# Create count of the number of epochs
score = mdl_1.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

train_acc = history_1.history['acc']
test_acc = history_1.history['val_acc']
plt_mdl_acc(epoch_count,train_acc,test_acc)
```




```
Test score: 0.335425318197847
Test accuracy: 0.9180683668062083
```



[2.2] Single Layer LSTM

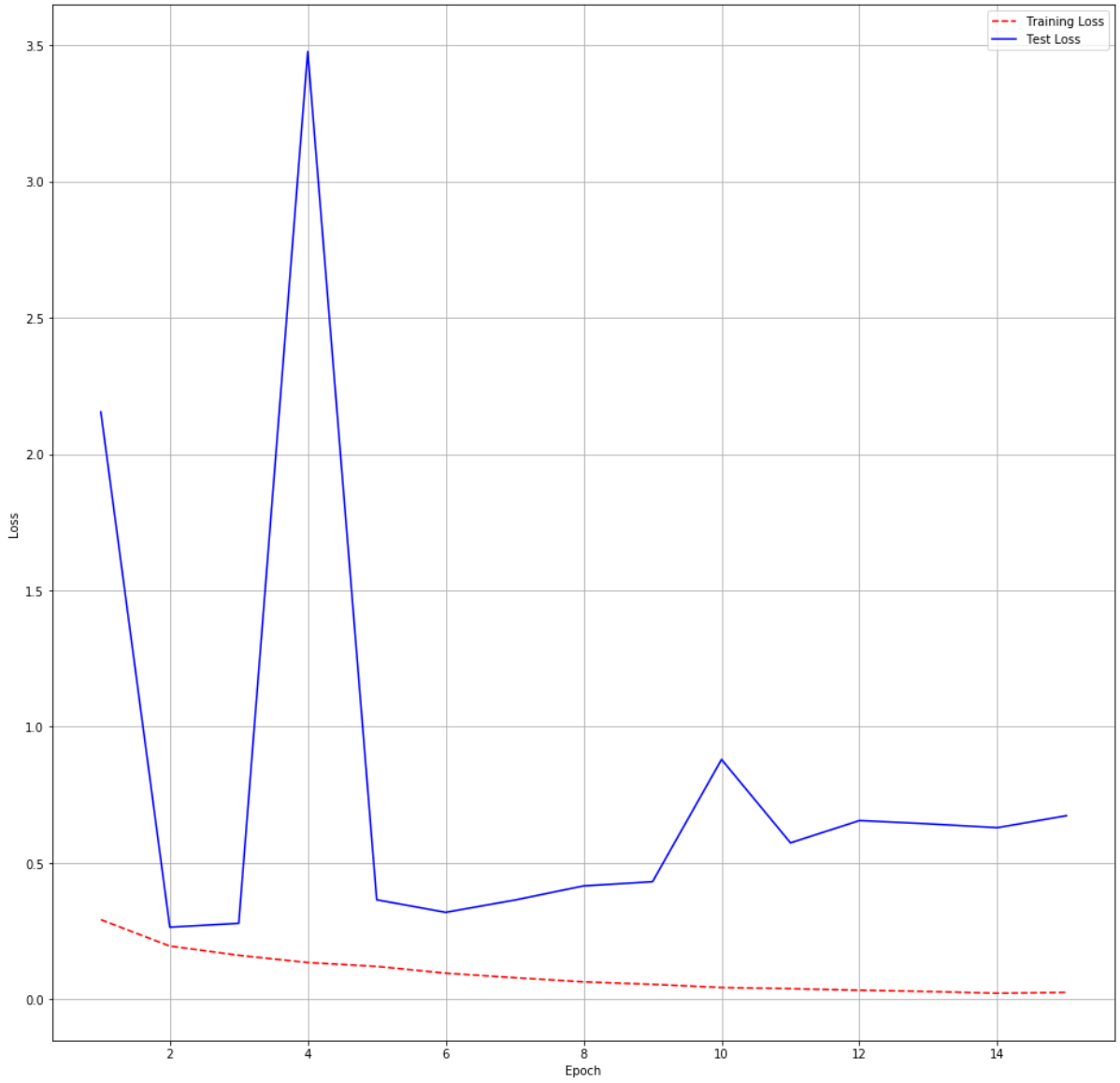
With Regularization and Batch Normalization and No Dropout

In [0]:
```
from keras.regularizers import l2
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 15
batch_size = 64
mdl_21 = Sequential()
mdl_21.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_
review_length))
mdl_21.add(LSTM(100,kernel_regularizer=l2(0.000001)))
mdl_21.add(BatchNormalization())
mdl_21.add(Dense(1, activation='sigmoid'))
mdl_21.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accu
racy'])
print(mdl_21.summary())
history_21 = mdl_21.fit(X_train, y_train, nb_epoch=no_epoch, batch_size=bat
ch_size,verbose=1,validation_data=(x_test,y_test) )
#Refer: https://datascience.stackexchange.com/questions/10615/number-of-par
ameters-in-an-lstm-model
```

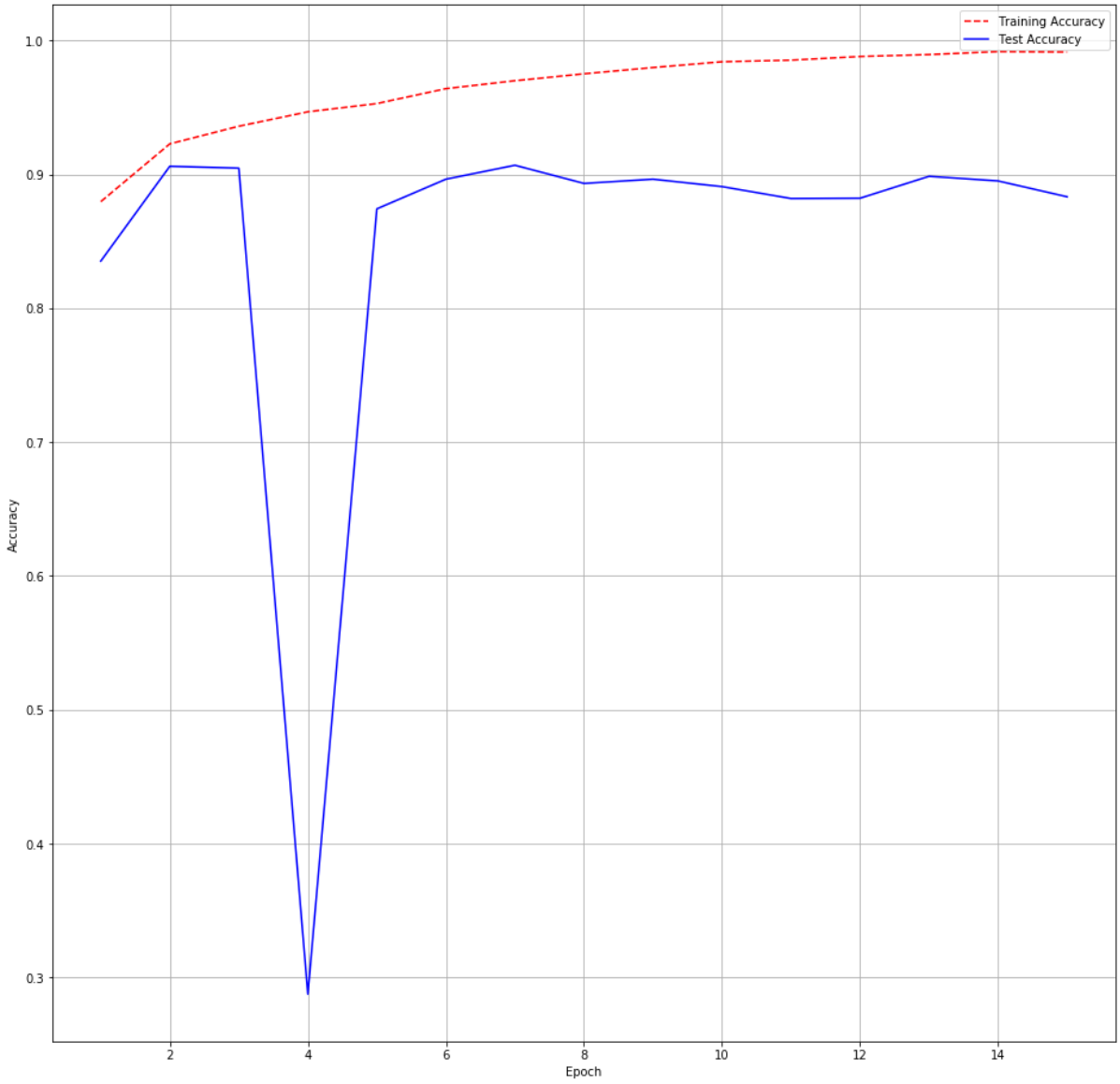
```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_6 (LSTM)                (None, 100)               53200
_________________________________________________________________
batch_normalization_5 (Batch (None, 100)               400
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 101
=================================================================
Total params: 213,733
Trainable params: 213,533
Non-trainable params: 200
_________________________________________________________________
None

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:14: UserWarnin
g: The `nb_epoch` argument in `fit` has been renamed `epochs`.
```

```
Train on 36856 samples, validate on 9215 samples
Epoch 1/15
36856/36856 [==============================] - 574s 16ms/step - loss: 0.291
9 - acc: 0.8796 - val_loss: 2.1543 - val_acc: 0.8352
Epoch 2/15
36856/36856 [==============================] - 550s 15ms/step - loss: 0.195
1 - acc: 0.9228 - val_loss: 0.2645 - val_acc: 0.9060
Epoch 3/15
36856/36856 [==============================] - 549s 15ms/step - loss: 0.161
3 - acc: 0.9359 - val_loss: 0.2786 - val_acc: 0.9046
Epoch 4/15
36856/36856 [==============================] - 549s 15ms/step - loss: 0.134
7 - acc: 0.9467 - val_loss: 3.4764 - val_acc: 0.2875
Epoch 5/15
36856/36856 [==============================] - 549s 15ms/step - loss: 0.120
5 - acc: 0.9529 - val_loss: 0.3649 - val_acc: 0.8742
Epoch 6/15
36856/36856 [==============================] - 550s 15ms/step - loss: 0.095
8 - acc: 0.9640 - val_loss: 0.3189 - val_acc: 0.8964
Epoch 7/15
36856/36856 [==============================] - 559s 15ms/step - loss: 0.079
1 - acc: 0.9699 - val_loss: 0.3639 - val_acc: 0.9068
Epoch 8/15
36856/36856 [==============================] - 553s 15ms/step - loss: 0.063
8 - acc: 0.9751 - val_loss: 0.4159 - val_acc: 0.8932
Epoch 9/15
36856/36856 [==============================] - 552s 15ms/step - loss: 0.054
9 - acc: 0.9798 - val_loss: 0.4313 - val_acc: 0.8964
Epoch 10/15
36856/36856 [==============================] - 554s 15ms/step - loss: 0.043
1 - acc: 0.9841 - val_loss: 0.8797 - val_acc: 0.8908
Epoch 11/15
36856/36856 [==============================] - 567s 15ms/step - loss: 0.039
0 - acc: 0.9853 - val_loss: 0.5739 - val_acc: 0.8819
Epoch 12/15
36856/36856 [==============================] - 570s 15ms/step - loss: 0.033
1 - acc: 0.9880 - val_loss: 0.6556 - val_acc: 0.8821
Epoch 13/15
36856/36856 [==============================] - 570s 15ms/step - loss: 0.028
5 - acc: 0.9895 - val_loss: 0.6435 - val_acc: 0.8985
Epoch 14/15
36856/36856 [==============================] - 563s 15ms/step - loss: 0.022
2 - acc: 0.9917 - val_loss: 0.6295 - val_acc: 0.8951
Epoch 15/15
36856/36856 [==============================] - 560s 15ms/step - loss: 0.025
3 - acc: 0.9914 - val_loss: 0.6733 - val_acc: 0.8832
```

In [0]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt

epoch_count = list(range(1,no_epoch+1))

# Get training and test loss histories
training_loss = history_21.history['loss']
test_loss = history_21.history['val_loss']

# Create count of the number of epochs
score = mdl_21.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

# Get training and test loss histories
training_aucrcy = history_21.history['acc']
test_aucrcy = history_21.history['val_acc']
plt_mdl_acc(epoch_count, training_aucrcy, test_aucrcy)
```




```
Test score: 0.6733268639977316
Test accuracy: 0.8832338577563906
```



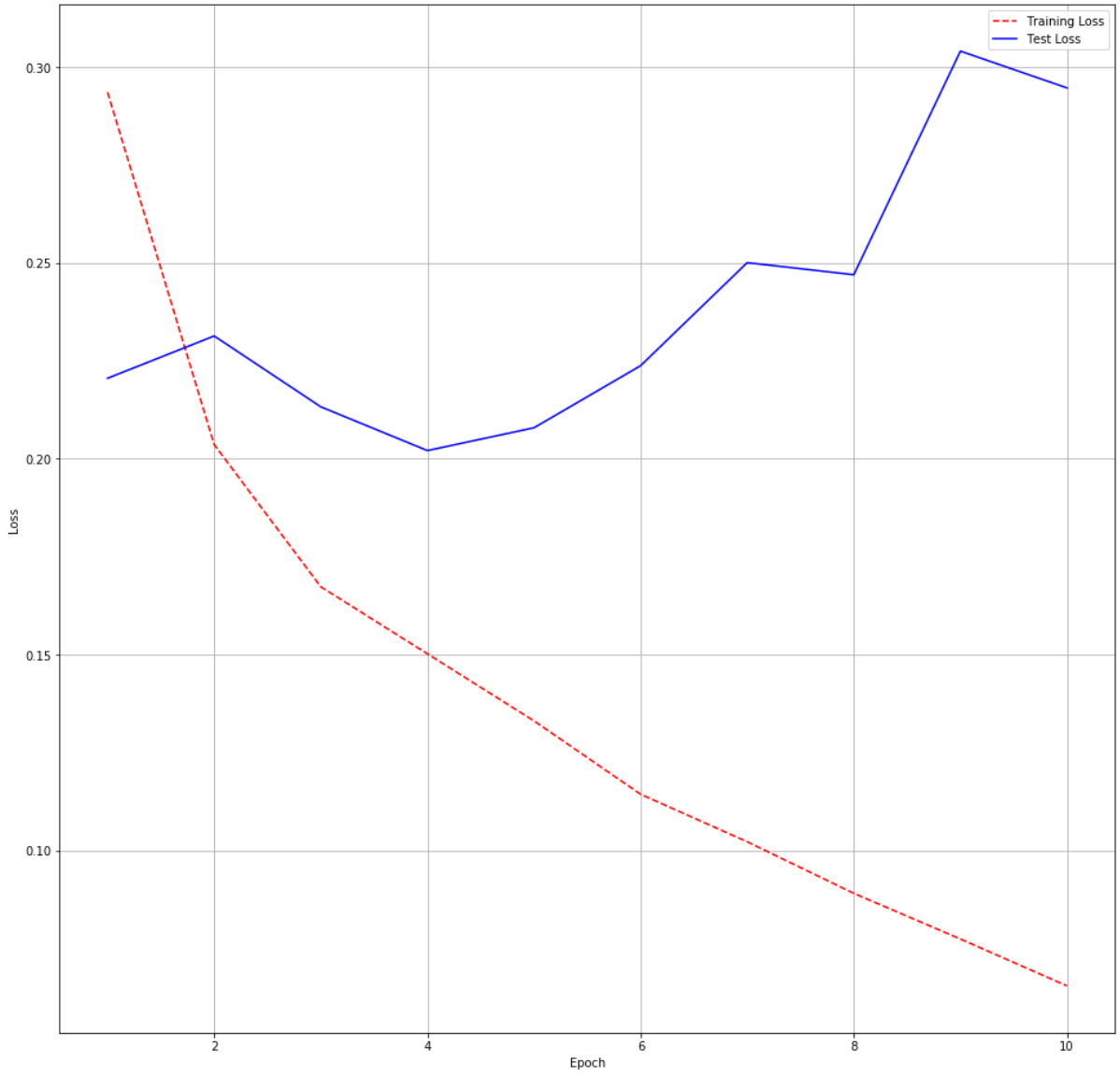
[3]Multi Layered LSTM

[3.1] No Dropout and Batch normalization

In [0]:
```
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 10
batch_size = 64
mdl_3 = Sequential()
mdl_3.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
mdl_3.add(LSTM(100,return_sequences=True))
mdl_3.add(LSTM(64,return_sequences=False))
mdl_3.add(Dense(1, activation='sigmoid'))
mdl_3.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accur
acy'])
print(mdl_3.summary())
history_3 = mdl_3.fit(X_train, y_train, nb_epoch=no_epoch, batch_size=batch
_size,verbose=1,validation_data=(x_test,y_test) )
#Refer: https://datascience.stackexchange.com/questions/10615/number-of-par
ameters-in-an-lstm-model
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_5 (LSTM)                (None, 600, 100)          53200
_________________________________________________________________
lstm_6 (LSTM)                (None, 64)                42240
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 255,537
Trainable params: 255,537
Non-trainable params: 0
_________________________________________________________________
None

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:12: UserWarnin
g: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  if sys.path[0] == '':

Train on 36856 samples, validate on 9215 samples
Epoch 1/10
36856/36856 [==============================] - 1028s 28ms/step - loss: 0.29
36 - acc: 0.8840 - val_loss: 0.2206 - val_acc: 0.9105
Epoch 2/10
36856/36856 [==============================] - 1000s 27ms/step - loss: 0.20
36 - acc: 0.9179 - val_loss: 0.2313 - val_acc: 0.9161
Epoch 3/10
36856/36856 [==============================] - 1031s 28ms/step - loss: 0.16
73 - acc: 0.9371 - val_loss: 0.2133 - val_acc: 0.9203
Epoch 4/10
36856/36856 [==============================] - 1031s 28ms/step - loss: 0.15
02 - acc: 0.9442 - val_loss: 0.2021 - val_acc: 0.9214
Epoch 5/10
36856/36856 [==============================] - 1028s 28ms/step - loss: 0.13
31 - acc: 0.9508 - val_loss: 0.2079 - val_acc: 0.9219
Epoch 6/10
36856/36856 [==============================] - 1032s 28ms/step - loss: 0.11
44 - acc: 0.9592 - val_loss: 0.2238 - val_acc: 0.9170
Epoch 7/10
36856/36856 [==============================] - 1033s 28ms/step - loss: 0.10
23 - acc: 0.9635 - val_loss: 0.2501 - val_acc: 0.9134
Epoch 8/10
36856/36856 [==============================] - 1030s 28ms/step - loss: 0.08
91 - acc: 0.9691 - val_loss: 0.2470 - val_acc: 0.9114
Epoch 9/10
36856/36856 [==============================] - 1031s 28ms/step - loss: 0.07
74 - acc: 0.9735 - val_loss: 0.3040 - val_acc: 0.9094
Epoch 10/10
36856/36856 [==============================] - 1030s 28ms/step - loss: 0.06
55 - acc: 0.9787 - val_loss: 0.2947 - val_acc: 0.9139
```

In [0]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt

epoch_count = list(range(1,no_epoch+1))

# Get training and test loss histories
training_loss = history_3.history['loss']
test_loss = history_3.history['val_loss']

# Create count of the number of epochs
score = mdl_3.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])
```




```
Test score: 0.29465569111069323
Test accuracy: 0.9139446553625106
```

[3.2] Multi Layered LSTM

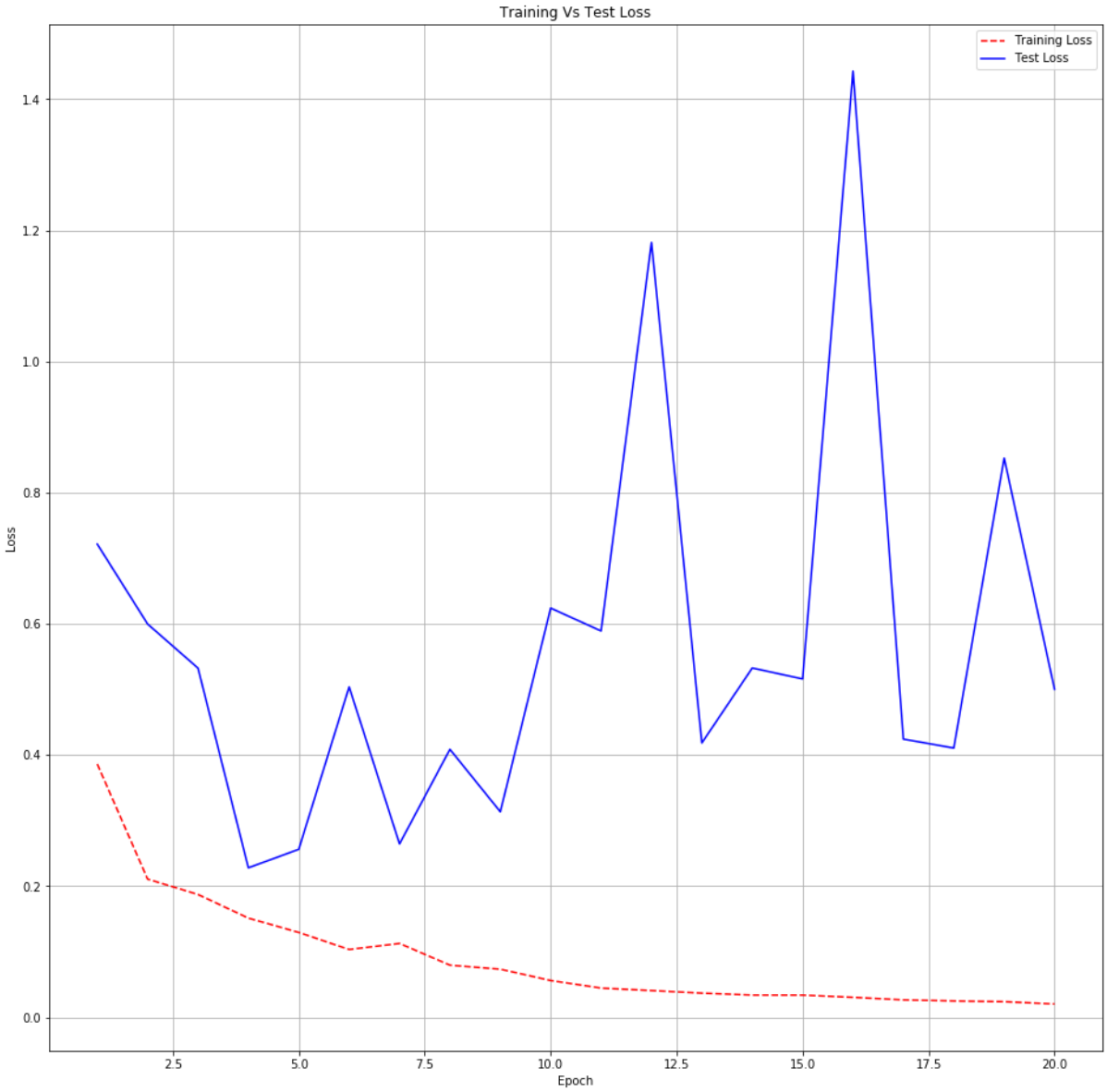
With Dropout rate = 0.2, Batch normalization and L2 regularization

In [16]:
```
# create the model
embedding_vecor_length = 32
top_words = 5000
no_epoch = 20
batch_size = 64
mdl_6 = Sequential()
mdl_6.add(Embedding(top_words+1, embedding_vecor_length, input_length=max_r
eview_length))
mdl_6.add(LSTM(100,return_sequences=True,kernel_regularizer=l2(0.000001)))
mdl_6.add(Dropout(rate=(0.2)))
mdl_6.add(LSTM(64,return_sequences=True,kernel_regularizer=l2(0.000001)))
mdl_6.add(BatchNormalization())
mdl_6.add(LSTM(32,return_sequences=False,kernel_regularizer=l2(0.000001)))
mdl_6.add(Dropout(rate=(0.2)))
mdl_6.add(BatchNormalization())
mdl_6.add(Dense(1, activation='sigmoid'))
mdl_6.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accur
acy'])
print(mdl_6.summary())
#tensorboard_callback = keras.callbacks.TensorBoard(log_dir=logs_base_dir)
history_6 = mdl_6.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_s
ize,verbose=1,validation_data=(x_test,y_test) )
%matplotlib inline
import keras
from matplotlib import pyplot as plt

epoch_count = list(range(1,no_epoch+1))

# Get training and test loss histories
training_loss = history_6.history['loss']
test_loss = history_6.history['val_loss']

# Create count of the number of epochs
score = mdl_6.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

# Get training and test loss histories
training_aucrcy = history_6.history['acc']
test_aucrcy = history_6.history['val_acc']
plt_mdl_acc(epoch_count, training_aucrcy, test_aucrcy)
#history_6 = mdl_6.fit(X_train, y_train, epochs=no_epoch, batch_size=batch_
size,verbose=1,validation_data=(x_test,y_test),callbacks=[tensorboard_callb
ack] )
#Refer: https://datascience.stackexchange.com/questions/10615/number-of-par
ameters-in-an-lstm-model
```

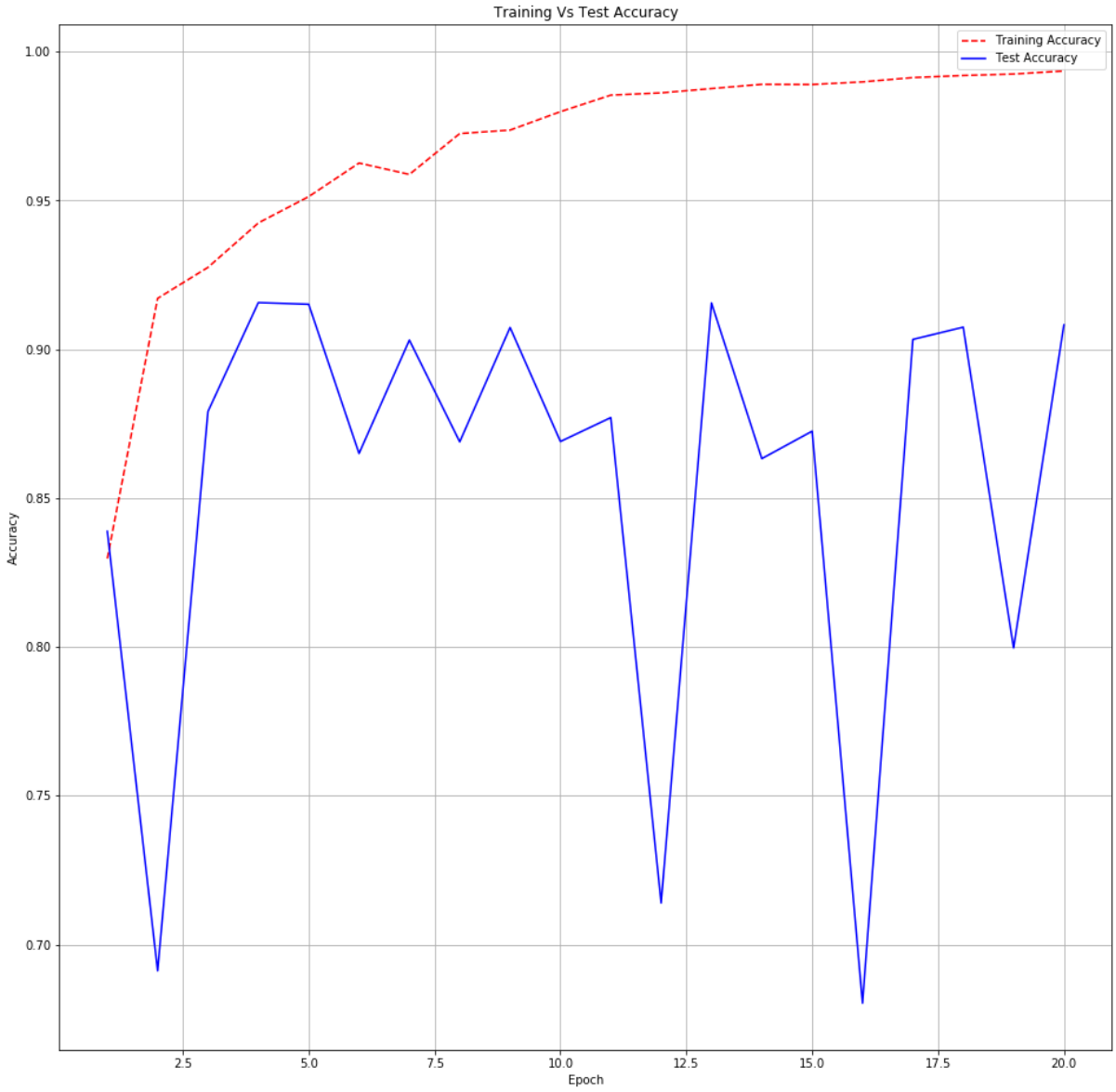
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_4 (Embedding)      (None, 600, 32)           160032
_________________________________________________________________
lstm_6 (LSTM)                (None, 600, 100)          53200
_________________________________________________________________
dropout_4 (Dropout)          (None, 600, 100)          0
_________________________________________________________________
lstm_7 (LSTM)                (None, 600, 64)           42240
_________________________________________________________________
batch_normalization_5 (Batch (None, 600, 64)           256
_________________________________________________________________
lstm_8 (LSTM)                (None, 32)                12416
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
batch_normalization_6 (Batch (None, 32)                128
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 268,305
Trainable params: 268,113
Non-trainable params: 192
_________________________________________________________________
None
Train on 36856 samples, validate on 9215 samples
Epoch 1/20
36856/36856 [==============================] - 1328s 36ms/step - loss: 0.38
63 - acc: 0.8297 - val_loss: 0.7217 - val_acc: 0.8388
Epoch 2/20
36856/36856 [==============================] - 1347s 37ms/step - loss: 0.21
07 - acc: 0.9171 - val_loss: 0.5997 - val_acc: 0.6912
Epoch 3/20
36856/36856 [==============================] - 1324s 36ms/step - loss: 0.18
72 - acc: 0.9275 - val_loss: 0.5324 - val_acc: 0.8791
Epoch 4/20
36856/36856 [==============================] - 1313s 36ms/step - loss: 0.15
12 - acc: 0.9425 - val_loss: 0.2279 - val_acc: 0.9157
Epoch 5/20
36856/36856 [==============================] - 1318s 36ms/step - loss: 0.12
94 - acc: 0.9513 - val_loss: 0.2561 - val_acc: 0.9151
Epoch 6/20
36856/36856 [==============================] - 1332s 36ms/step - loss: 0.10
33 - acc: 0.9626 - val_loss: 0.5037 - val_acc: 0.8650
Epoch 7/20
36856/36856 [==============================] - 1329s 36ms/step - loss: 0.11
25 - acc: 0.9588 - val_loss: 0.2643 - val_acc: 0.9031
Epoch 8/20
36856/36856 [==============================] - 1320s 36ms/step - loss: 0.07
94 - acc: 0.9725 - val_loss: 0.4086 - val_acc: 0.8689
Epoch 9/20
36856/36856 [==============================] - 1313s 36ms/step - loss: 0.07
33 - acc: 0.9737 - val_loss: 0.3133 - val_acc: 0.9073
Epoch 10/20
36856/36856 [==============================] - 1312s 36ms/step - loss: 0.05
```

```
60 - acc: 0.9798 - val_loss: 0.6239 - val_acc: 0.8690
Epoch 11/20
36856/36856 [==============================] - 1313s 36ms/step - loss: 0.04
44 - acc: 0.9854 - val_loss: 0.5891 - val_acc: 0.8770
Epoch 12/20
36856/36856 [==============================] - 1313s 36ms/step - loss: 0.04
07 - acc: 0.9862 - val_loss: 1.1816 - val_acc: 0.7139
Epoch 13/20
36856/36856 [==============================] - 1312s 36ms/step - loss: 0.03
68 - acc: 0.9876 - val_loss: 0.4183 - val_acc: 0.9156
Epoch 14/20
36856/36856 [==============================] - 1311s 36ms/step - loss: 0.03
38 - acc: 0.9890 - val_loss: 0.5325 - val_acc: 0.8633
Epoch 15/20
36856/36856 [==============================] - 1313s 36ms/step - loss: 0.03
38 - acc: 0.9890 - val_loss: 0.5160 - val_acc: 0.8725
Epoch 16/20
36856/36856 [==============================] - 1312s 36ms/step - loss: 0.03
03 - acc: 0.9899 - val_loss: 1.4429 - val_acc: 0.6803
Epoch 17/20
36856/36856 [==============================] - 1311s 36ms/step - loss: 0.02
65 - acc: 0.9913 - val_loss: 0.4241 - val_acc: 0.9033
Epoch 18/20
36856/36856 [==============================] - 1312s 36ms/step - loss: 0.02
49 - acc: 0.9920 - val_loss: 0.4106 - val_acc: 0.9074
Epoch 19/20
36856/36856 [==============================] - 1317s 36ms/step - loss: 0.02
38 - acc: 0.9925 - val_loss: 0.8527 - val_acc: 0.7997
Epoch 20/20
36856/36856 [==============================] - 1315s 36ms/step - loss: 0.02
02 - acc: 0.9935 - val_loss: 0.5001 - val_acc: 0.9082
```

In [17]:
```
%matplotlib inline
import keras
from matplotlib import pyplot as plt

epoch_count = list(range(1,no_epoch+1))

# Get training and test loss histories
training_loss = history_6.history['loss']
test_loss = history_6.history['val_loss']

# Create count of the number of epochs
score = mdl_6.evaluate(x_test, y_test, verbose=0)
plt_mdl_res(epoch_count, training_loss, test_loss, score[0], score[1])

# Get training and test loss histories
training_aucrcy = history_6.history['acc']
test_aucrcy = history_6.history['val_acc']
plt_mdl_acc(epoch_count, training_aucrcy, test_aucrcy)
```




```
Test score: 0.5001427056665046
Test accuracy: 0.9081931633271411
```



[4] Conclusion

In [24]:
```
import tabulate
res_tab =[['Layer','Dimesnion \nOne','Epochs','Regularization \nl2','Batch
\nNormalization','Train \nLoss','Test \nLoss','Test \nAccuracy','Refer \nSe
ction'],
         [1,16,15,0.000001,'Yes',0.4417,0.51914,0.835,'2.1.2'],
         [1,100,15,0.000001,'Yes',0.0253,0.6733,0.8832,2.2]]
print(tabulate.tabulate(res_tab,tablefmt='fancy_grid'))
```

```
╒═══════╤═══════════╤══════════╤═══════════════╤═══════════════╤═══════╤
═══════╤═══════════╤═══════════╕
│ Layer │ Dimesnion │ Epochs   │ Regularization │ Batch         │ Train │
Test   │ Test      │ Refer     │
│       │ One       │          │ l2             │ Normalization │ Loss  │
Loss   │ Accuracy  │ Section   │
├───────┼───────────┼──────────┼───────────────┼───────────────┼───────┼─
───────┼───────────┼───────────┤
│ 1     │ 16        │ 15       │ 1e-06          │ Yes           │ 0.4417 │
0.51914 │ 0.835     │ 2.1.2     │
├───────┼───────────┼──────────┼───────────────┼───────────────┼───────┼─
───────┼───────────┼───────────┤
│ 1     │ 100       │ 15       │ 1e-06          │ Yes           │ 0.0253 │
0.6733  │ 0.8832    │ 2.2       │
╘═══════╧═══════════╧══════════╧═══════════════╧═══════════════╧═══════╧
═══════╧═══════════╧═══════════╛
```

In [1]:
```
import tabulate
res_tab2 =[['Layer','Dimesnion \nOne','Dimension \nTwo','Epochs','Batch \nN
ormalization','Train \nLoss','Test \nLoss','Test \nAccuracy','Refer \nSecti
on'],
         [1,100,64,10,'No',0.0655,0.2947,0.9139,'3.1']]
print(tabulate.tabulate(res_tab2,tablefmt='fancy_grid'))
```

```
╒═══════╤═══════════╤═══════════╤══════════╤═══════════════╤═══════╤═══════
═══╤═══════════╤═══════════╕
│ Layer │ Dimesnion │ Dimension │ Epochs   │ Batch         │ Train │ Test
│ Test      │ Refer     │
│       │ One       │ Two       │          │ Normalization │ Loss  │ Loss
│ Accuracy  │ Section   │
├───────┼───────────┼───────────┼──────────┼───────────────┼───────┼──────
─┼───────────┼───────────┤
│ 1     │ 100       │ 64        │ 10       │ No            │ 0.0655 │ 0.294
7 │ 0.9139    │ 3.1       │
╘═══════╧═══════════╧═══════════╧══════════╧═══════════════╧═══════╧══════
═╧═══════════╧═══════════╛
```

In [26]:
```
import tabulate
res_tab3 =[['Layer','Dimesnion \nOne','Dimension \nTwo','Dimension \nThree'
,'Epochs','Regularization \nl2','Drop-out \nRate','Batch \nNormalization',
'Train \nLoss','Test \nLoss','Test \nAccuracy','Refer \nSection'],
        [1,100,64,32,20,0.000001,0.2,'Yes',0.0368,0.4183,0.9081,'3.2']]
print(tabulate.tabulate(res_tab3,tablefmt='fancy_grid'))
```

```
╒═══════╤═══════════╤═══════════╤═══════════╤══════════╤═══════════════╤
═══════════╤═══════════╤═══════╤═══════╤══════════╤═══════════╕
│ Layer │ Dimesnion │ Dimension │ Dimension │ Epochs   │ Regularization │
Drop-out  │ Batch     │ Train │ Test  │ Test     │ Refer     │
│       │ One       │ Two       │ Three     │          │ l2             │
Rate      │ Normalization │ Loss  │ Loss  │ Accuracy │ Section   │
├───────┼───────────┼───────────┼───────────┼──────────┼───────────────┼
───────────┼───────────┼───────┼───────┼──────────┼───────────┤
│ 1     │ 100       │ 64        │ 32        │ 20       │ 1e-06         │
0.2       │ Yes       │ 0.0368 │ 0.4183 │ 0.9081   │ 3.2       │
╘═══════╧═══════════╧═══════════╧═══════════╧══════════╧═══════════════╧
═══════════╧═══════════╧═══════╧═══════╧══════════╧═══════════╛
```

In all the above models graphs, it can be observed that the test loss is not increasing after decreasing . This shows that the models are not overfitting the data.