LSTM on Amazon fine food reviews data

In [ ]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re

import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
from keras.layers import Dropout

#Importing Cleaned & Deduped dataset
```

```
# using the SQLite Table to read data.
con = sqlite3.connect('C:/Users/deepak/Documents/Applied AI assignments/3. Tsne on Amazon fine food/final.sqlite
```

In [5]:
```
Data = pd.read_sql_query(""" SELECT * FROM Reviews""", con)
```

In [6]:
```
Data['Score'].value_counts()
```

Out[6]:
```
positive    307061
negative     57110
Name: Score, dtype: int64
```

```
In [7]: Data.head(5)
```

Out[7]:

lex	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary
'06	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	positive	939340800	EVERY book is educational
;88	150506	0006641040	A2IW4PEEKO2R0U	Tracy	1	1	positive	1194739200	Love the book, miss the hard cover version
;89	150507	0006641040	A1S4A3IQ2MU7V4	sally sue "sally sue"	1	1	positive	1191456000	chicken soup with rice months
;90	150508	0006641040	AZGXZ2UUK6X	Catherine Hallberg " (Kate)"	1	1	positive	1076025600	a good swingy rhythm for reading aloud
;91	150509	0006641040	A3CMRKGE0P909G	Teresa	3	4	positive	1018396800	A great way to learn the months

In [8]:

```
#Sorting the data ascending order
Data=Data.sort_values("Time",ascending = True)

text = Data['CleanedText'].values
y = Data['Score']
```

In [9]:

```
# Finding all words in the vocabulary
count_vect = CountVectorizer()
count_vect.fit(text)

vocabulary = count_vect.get_feature_names()
print(len(vocabulary))
```

```
71624
```

In [10]:

```
# Code reference - https://stackoverflow.com/questions/4088265/sorted-word-frequency-count-using-python
from collections import Counter
cnt = Counter()
for sent in text:
  for word in sent.split():
    cnt[word] += 1
```

In [11]:
```
cnt
```

Out[11]:
```
Counter({'witti': 11,
         'littl': 51736,
         'book': 2053,
         'make': 84947,
         'son': 7969,
         'laugh': 534,
         'loud': 318,
         'recit': 13,
         'car': 1961,
         'drive': 1699,
         'along': 5326,
         'alway': 23391,
         'sing': 226,
         'refrain': 49,
         'hes': 2956,
         'learn': 3231,
         'whale': 25,
         'india': 874,
         'droop': 18,
         'love': 115870,
```

In [12]:
```
#Sorting cnt in descending order
import operator
sort_cnt = sorted(cnt.items(), key=operator.itemgetter(1), reverse=True)[:5000]
```

In [13]:
```
sort_cnt
```

Out[13]:
```
[('like', 171759),
 ('tast', 163632),
 ('flavor', 129199),
 ('good', 127807),
 ('product', 119251),
 ('use', 119190),
 ('one', 117295),
 ('love', 115870),
 ('great', 109772),
 ('tri', 104544),
 ('tea', 95553),
 ('coffe', 93530),
 ('get', 85911),
 ('make', 84947),
 ('food', 77556),
 ('would', 73540),
 ('buy', 67950),
 ('time', 65063),
 ('realli', 62110),
```

In [14]:
```
len(sort_cnt)
```

Out[14]:
```
5000
```

In [15]:
```
# Assigning Index to top most sorted words
word_index_lookup = dict()
i = 1
# https://stackoverflow.com/questions/5466618/too-many-values-to-unpack-iterating-over-a-dict-key-string-value-l
for word,frequency in sort_cnt:
    word_index_lookup[word] = i
    i += 1
```

In [16]:
```
word_index_lookup
```

Out[16]:
```
{'like': 1,
 'tast': 2,
 'flavor': 3,
 'good': 4,
 'product': 5,
 'use': 6,
 'one': 7,
 'love': 8,
 'great': 9,
 'tri': 10,
 'tea': 11,
 'coffe': 12,
 'get': 13,
 'make': 14,
 'food': 15,
 'would': 16,
 'buy': 17,
 'time': 18,
 'realli': 19,
```

In [17]:
```
#Adding index column to the text
def apply_text_index(row):
    holder = []
    for word in row['CleanedText'].split():
        if word in word_index_lookup:
            holder.append(word_index_lookup[word])
        else:
            holder.append(0)
    return holder
```

In [18]:
```
Data['CleanedText_Index'] = Data.apply(lambda row: apply_text_index(row),axis=1)
```

In [19]:
```
Data['Score'] = Data['Score'].map(lambda x : 1 if x == 'positive' else 0)
```

In [20]:
```
Data
```

serId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	CleanedTex
IDDL	shari zychinski	0	0	1	939340800	EVERY book is educational	this witty little book makes my son laugh at l...	witti littl boo make so laugh lou recit car.
:7NR	Nicholas A Mesiano	2	2	1	940809600	This whole series is great way to spend time w...	I can remember seeing the show when it aired o...	rememb se show a televis yea ago chil sis.
)EG5	Elizabeth Medina	0	0	1	944092800	Entertainingl Funny!	Beetlejuice is a well written movie ..... ever...	beetleju well write movi everyt excel act.
CGM	Vincent P. Ross	1	2	1	944438400	A modern day fairy tale	A twist of rumplestiskin captured on film, sta...	twis rumplestiski captur filr star michae

In [21]:
```
# 70-30 split
x = Data['CleanedText_Index'].values
y = Data['Score']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)
```

In [22]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_train.shape)
```

```
(254919,)
(109252,)
(254919,)
(254919,)
```

In [23]:
```
x_train
```

Out[23]:
```
array([list([8, 4313, 12, 0, 229, 98, 112, 229, 12, 718, 203]),
       list([62, 48, 18, 21, 195, 211, 50, 9, 204, 98, 3228, 46, 1211, 159, 55, 105, 211, 1012, 50, 688, 189,
50, 76, 286, 2610, 110, 149, 240, 749, 46, 1, 3177, 295, 3228, 143, 240, 0, 1009, 1, 3499, 180, 1561, 311, 19,
1, 1012, 211, 75]),
       list([205, 7, 118, 716, 1203, 1181, 3164, 512, 387, 613, 257, 743, 468, 200, 402, 1203]),
       ...,
       list([471, 14, 1489, 215, 177, 2, 19, 4, 373, 40, 106, 406, 782, 2375, 373, 52, 471, 1489, 1116, 325, 6
42, 3, 494, 1, 165, 925, 1307, 296, 215, 745, 2, 1, 200, 53, 500, 313, 643, 272, 375, 1367, 146, 254, 108, 37,
1965, 4195]),
       list([1817, 834, 222, 0, 43, 182, 18, 375, 167, 112, 9, 17, 85, 1701, 120]),
       list([1, 72, 54, 4153, 1010, 109, 643, 8, 22, 397, 9, 83, 68, 90, 255, 10, 995, 2125, 549, 672, 427, 70
6, 1720, 9, 170, 460, 32, 9, 706, 1748, 68, 1, 68, 1, 702, 500, 479, 4, 16, 670, 340, 14, 22, 623, 118, 18])],
      dtype=object)
```

In [24]:
```
# max words in single sentence of cleaned text to apply padding
Data['number_of_words'] = Data.CleanedText.apply(lambda x: len(x.split()))
sort_data=Data.sort_values(by='number_of_words', axis=0, ascending=False)
```

In [25]:
```
sort_data.head(1)
```

Out[25]:

ssNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	CleanedText	CleanedText_Ind
4	4	1	1323993600	SEARCHING FOR A PET APPETITE ENHANCER?	**...	read updat first thank june august activ anywa...	[232, 1247, 48, 17 2673, 2992, 119 574, 50

In [26]:

```
max_review_length = 1355 # set as per the max no of words in single sentence
x_train = sequence.pad_sequences(x_train, maxlen=max_review_length)
x_test = sequence.pad_sequences(x_test, maxlen=max_review_length)

print("Total number words present in first review after padding:\n",len(x_train[0]))
print()
print("List of word indexes present in first review padding:\n", x_train[0])
print()
```

```
Total number words present in first review after padding:
 1355

List of word indexes present in first review padding:
 [  0   0   0 ...  12 718 203]
```

In [40]:

```
#function for plotting train v/s validation loss
def plt_dynamic(x, vy, ty):
  plt.figure(figsize=(10,5))
  plt.plot(x, vy, 'b', label="Validation Loss")
  plt.plot(x, ty, 'r', label="Train Loss")
  plt.xlabel('Epochs')
  plt.ylabel('Binary Crossentropy Loss')
  plt.title('\nBinary Crossentropy Loss VS Epochs')
  plt.legend()
  plt.grid()
  plt.show()
```

In [39]:

```
from __future__ import print_function
import numpy

from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
import numpy

from keras.wrappers.scikit_learn import KerasClassifier
from keras.optimizers import SGD
from keras.constraints import maxnorm
from keras import Sequential
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from keras.models import Sequential,Model
from keras.layers import LSTM, Dense, Bidirectional, Input,Dropout,BatchNormalization, CuDNNGRU, CuDNNLSTM

from keras import backend as K
from keras.engine.topology import Layer
from keras import initializers, regularizers, constraints

# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
```

Trying different LSTM Layers

1. 1 LSTM LAYER

In [37]:

```
# create the model
embedding_vecor_length = 32
model_1 = Sequential()
model_1.add(Embedding(len(vocabulary), embedding_vecor_length, input_length=max_review_length))
model_1.add(Dropout(0.5))
model_1.add(CuDNNLSTM(256))
model_1.add(Dropout(0.3))
model_1.add(Dense(1, activation='sigmoid'))
print(model_1.summary())

# Compiling the model
model_1.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_6 (Embedding)      (None, 1355, 32)          2291968
_________________________________________________________________
dropout_11 (Dropout)         (None, 1355, 32)          0
_________________________________________________________________
cu_dnnlstm_5 (CuDNNLSTM)     (None, 256)               296960
_________________________________________________________________
dropout_12 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 257
=================================================================
Total params: 2,589,185
Trainable params: 2,589,185
Non-trainable params: 0
_________________________________________________________________
None
```

In [38]:

```
# Fitting the data to the model
history = model_1.fit(x_train, y_train, nb_epoch=10, batch_size=512 ,verbose=1,validation_data=(x_test, y_test))
```

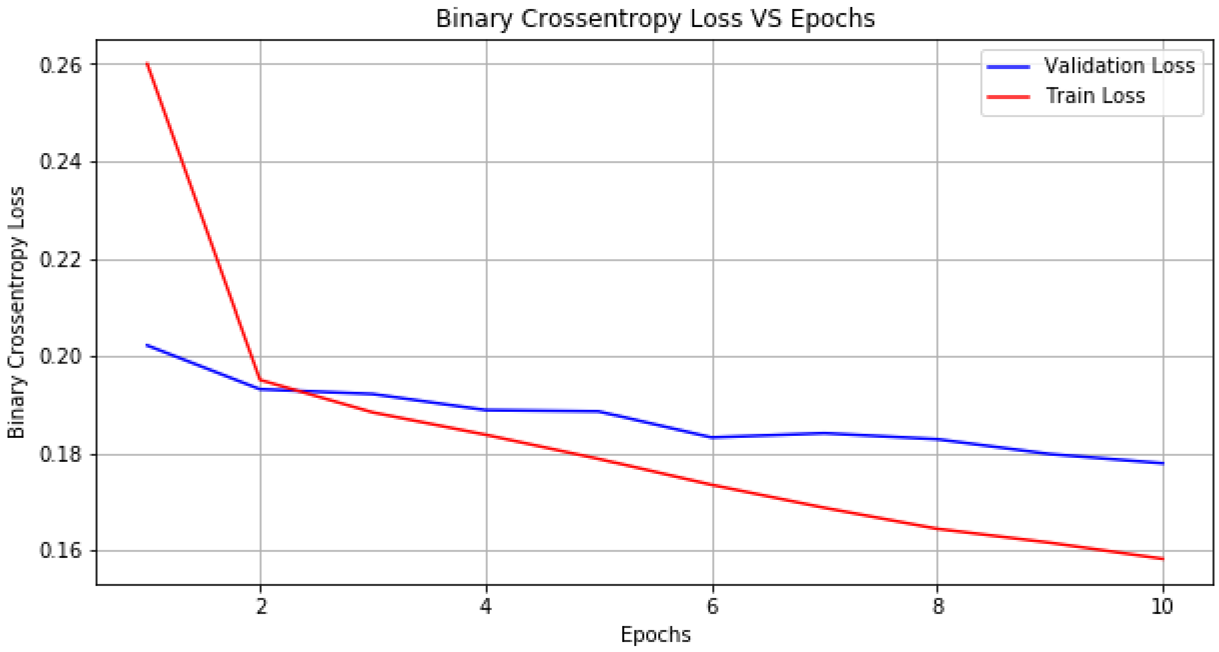
```
acc: 0.93 - ETA: 30s - loss: 0.1572 - acc: 0.93 - ETA: 29s - loss: 0.1573 - acc: 0.93 - ETA: 29s - loss: 0.1
573 - acc: 0.93 - ETA: 28s - loss: 0.1573 - acc: 0.93 - ETA: 27s - loss: 0.1572 - acc: 0.93 - ETA: 27s - los
s: 0.1572 - acc: 0.93 - ETA: 26s - loss: 0.1572 - acc: 0.93 - ETA: 26s - loss: 0.1573 - acc: 0.93 - ETA: 25s
- loss: 0.1572 - acc: 0.93 - ETA: 24s - loss: 0.1573 - acc: 0.93 - ETA: 24s - loss: 0.1573 - acc: 0.93 - ET
A: 23s - loss: 0.1573 - acc: 0.93 - ETA: 23s - loss: 0.1573 - acc: 0.93 - ETA: 22s - loss: 0.1572 - acc: 0.9
3 - ETA: 22s - loss: 0.1572 - acc: 0.93 - ETA: 21s - loss: 0.1573 - acc: 0.93 - ETA: 20s - loss: 0.1573 - ac
c: 0.93 - ETA: 20s - loss: 0.1572 - acc: 0.93 - ETA: 19s - loss: 0.1572 - acc: 0.93 - ETA: 19s - loss: 0.157
2 - acc: 0.93 - ETA: 18s - loss: 0.1574 - acc: 0.93 - ETA: 17s - loss: 0.1575 - acc: 0.93 - ETA: 17s - loss:
0.1575 - acc: 0.93 - ETA: 16s - loss: 0.1575 - acc: 0.93 - ETA: 16s - loss: 0.1575 - acc: 0.93 - ETA: 15s -
loss: 0.1575 - acc: 0.93 - ETA: 15s - loss: 0.1576 - acc: 0.93 - ETA: 14s - loss: 0.1576 - acc: 0.93 - ETA:
13s - loss: 0.1577 - acc: 0.93 - ETA: 13s - loss: 0.1576 - acc: 0.93 - ETA: 12s - loss: 0.1576 - acc: 0.93 -
ETA: 12s - loss: 0.1576 - acc: 0.93 - ETA: 11s - loss: 0.1577 - acc: 0.93 - ETA: 10s - loss: 0.1577 - acc:
0.93 - ETA: 10s - loss: 0.1577 - acc: 0.93 - ETA: 9s - loss: 0.1577 - acc: 0.9377 - ETA: 9s - loss: 0.1578 -
acc: 0.937 - ETA: 8s - loss: 0.1578 - acc: 0.937 - ETA: 8s - loss: 0.1578 - acc: 0.937 - ETA: 7s - loss: 0.1
579 - acc: 0.937 - ETA: 6s - loss: 0.1578 - acc: 0.937 - ETA: 6s - loss: 0.1578 - acc: 0.937 - ETA: 5s - los
s: 0.1578 - acc: 0.937 - ETA: 5s - loss: 0.1579 - acc: 0.937 - ETA: 4s - loss: 0.1580 - acc: 0.937 - ETA: 4s
- loss: 0.1581 - acc: 0.937 - ETA: 3s - loss: 0.1581 - acc: 0.937 - ETA: 2s - loss: 0.1582 - acc: 0.937 - ET
A: 2s - loss: 0.1582 - acc: 0.937 - ETA: 1s - loss: 0.1582 - acc: 0.937 - ETA: 1s - loss: 0.1582 - acc: 0.93
7 - ETA: 0s - loss: 0.1582 - acc: 0.937 - 328s 1ms/step - loss: 0.1583 - acc: 0.9375 - val_loss: 0.1779 - va
l_acc: 0.9299
```

In [42]:

```
x = list(range(1,11))

# Validation loss
vy = history.history['val_loss']
# Training loss
ty = history.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```



2. 2x2 LSTM LAYER

In [45]:

```
# create the model
embedding_vecor_length = 32
model_2 = Sequential()
model_2.add(Embedding(len(vocabulary), embedding_vecor_length, input_length=max_review_length))

# Add batch normalization
model_2.add(BatchNormalization())

model_2.add(Dropout(0.8))
#First layer
model_2.add(CuDNNLSTM(256,return_sequences=True))
model_2.add(Dropout(0.6))

#Second layer
model_2.add(CuDNNLSTM(128))
model_2.add(Dropout(0.5))


model_2.add(Dense(1, activation='sigmoid'))
print(model_2.summary())

# Compiling the model
model_2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_9 (Embedding)      (None, 1355, 32)          2291968   
_________________________________________________________________
batch_normalization_6 (Batch (None, 1355, 32)          128       
_________________________________________________________________
dropout_30 (Dropout)         (None, 1355, 32)          0         
_________________________________________________________________
cu_dnnlstm_20 (CuDNNLSTM)    (None, 1355, 256)         296960    
_________________________________________________________________
dropout_31 (Dropout)         (None, 1355, 256)         0         
_________________________________________________________________
cu_dnnlstm_21 (CuDNNLSTM)    (None, 128)               197632    
_________________________________________________________________
dropout_32 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 129       
```

```
=================================================================
Total params: 2,786,817
Trainable params: 2,786,753
Non-trainable params: 64
_________________________________________________________________
None
```

In [46]:

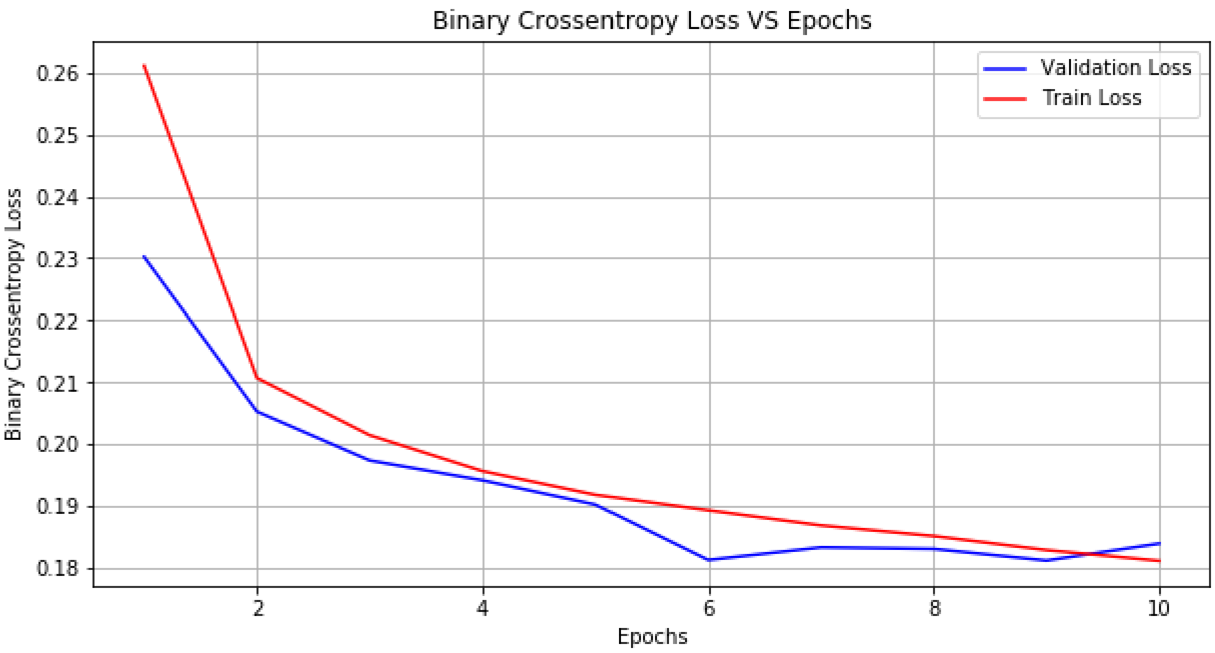
```
# Fitting the data to the model
history2 = model_2.fit(x_train, y_train, nb_epoch=10, batch_size=256 ,verbose=1,validation_data=(x_test, y_test)
```

```
acc: 0.92 - ETA: 29s - loss: 0.1813 - acc: 0.92 - ETA: 28s - loss: 0.1813 - acc: 0.92 - ETA: 28s - loss: 0.1
813 - acc: 0.92 - ETA: 27s - loss: 0.1813 - acc: 0.92 - ETA: 27s - loss: 0.1813 - acc: 0.92 - ETA: 26s - los
s: 0.1813 - acc: 0.92 - ETA: 25s - loss: 0.1813 - acc: 0.92 - ETA: 25s - loss: 0.1812 - acc: 0.92 - ETA: 24s
- loss: 0.1813 - acc: 0.92 - ETA: 24s - loss: 0.1814 - acc: 0.92 - ETA: 23s - loss: 0.1814 - acc: 0.92 - ET
A: 23s - loss: 0.1813 - acc: 0.92 - ETA: 22s - loss: 0.1814 - acc: 0.92 - ETA: 22s - loss: 0.1815 - acc: 0.9
2 - ETA: 21s - loss: 0.1814 - acc: 0.92 - ETA: 20s - loss: 0.1814 - acc: 0.92 - ETA: 20s - loss: 0.1814 - ac
c: 0.92 - ETA: 19s - loss: 0.1814 - acc: 0.92 - ETA: 19s - loss: 0.1814 - acc: 0.92 - ETA: 18s - loss: 0.181
4 - acc: 0.92 - ETA: 18s - loss: 0.1814 - acc: 0.92 - ETA: 17s - loss: 0.1814 - acc: 0.92 - ETA: 17s - loss:
0.1814 - acc: 0.92 - ETA: 16s - loss: 0.1814 - acc: 0.92 - ETA: 15s - loss: 0.1813 - acc: 0.92 - ETA: 15s -
loss: 0.1813 - acc: 0.92 - ETA: 14s - loss: 0.1813 - acc: 0.92 - ETA: 14s - loss: 0.1813 - acc: 0.92 - ETA:
13s - loss: 0.1813 - acc: 0.92 - ETA: 13s - loss: 0.1813 - acc: 0.92 - ETA: 12s - loss: 0.1813 - acc: 0.92 -
ETA: 12s - loss: 0.1813 - acc: 0.92 - ETA: 11s - loss: 0.1813 - acc: 0.92 - ETA: 10s - loss: 0.1813 - acc:
0.92 - ETA: 10s - loss: 0.1813 - acc: 0.92 - ETA: 9s - loss: 0.1812 - acc: 0.9286 - ETA: 9s - loss: 0.1813 -
acc: 0.928 - ETA: 8s - loss: 0.1813 - acc: 0.928 - ETA: 8s - loss: 0.1813 - acc: 0.928 - ETA: 7s - loss: 0.1
814 - acc: 0.928 - ETA: 7s - loss: 0.1814 - acc: 0.928 - ETA: 6s - loss: 0.1814 - acc: 0.928 - ETA: 5s - los
s: 0.1813 - acc: 0.928 - ETA: 5s - loss: 0.1813 - acc: 0.928 - ETA: 4s - loss: 0.1814 - acc: 0.928 - ETA: 4s
- loss: 0.1813 - acc: 0.928 - ETA: 3s - loss: 0.1813 - acc: 0.928 - ETA: 3s - loss: 0.1813 - acc: 0.928 - ET
A: 2s - loss: 0.1812 - acc: 0.928 - ETA: 2s - loss: 0.1812 - acc: 0.928 - ETA: 1s - loss: 0.1812 - acc: 0.92
8 - ETA: 0s - loss: 0.1811 - acc: 0.928 - ETA: 0s - loss: 0.1812 - acc: 0.928 - 630s 2ms/step - loss: 0.1812
- acc: 0.9286 - val_loss: 0.1840 - val_acc: 0.9266
```

In [47]:
```
x = list(range(1,11))

# Validation loss
vy = history2.history['val_loss']
# Training loss
ty = history2.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```



3. 3x3 LSTM LAYER

In [48]:

```
# create the model
embedding_vecor_length = 32
model_3 = Sequential()

model_3.add(Embedding(len(vocabulary), embedding_vecor_length, input_length=max_review_length))

# Add batch normalization
model_3.add(BatchNormalization())

model_3.add(Dropout(0.8))
#First layer
model_3.add(CuDNNLSTM(256,return_sequences=True))
model_3.add(Dropout(0.6))

#Second layer
model_3.add(CuDNNLSTM(128,return_sequences=True))
model_3.add(Dropout(0.5))

#Third layer
model_3.add(CuDNNLSTM(64))
model_3.add(Dropout(0.3))

model_3.add(Dense(1, activation='sigmoid'))
print(model_3.summary())

# Compiling the model
model_3.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_10 (Embedding)     (None, 1355, 32)          2291968
_________________________________________________________________
batch_normalization_7 (Batch (None, 1355, 32)          128
_________________________________________________________________
dropout_33 (Dropout)         (None, 1355, 32)          0
_________________________________________________________________
cu_dnnlstm_22 (CuDNNLSTM)    (None, 1355, 256)         296960
_________________________________________________________________
dropout_34 (Dropout)         (None, 1355, 256)         0
_________________________________________________________________
cu_dnnlstm_23 (CuDNNLSTM)    (None, 1355, 128)         197632
```

```
_________________________________________________________________
dropout_35 (Dropout)         (None, 1355, 128)         0
_________________________________________________________________
cu_dnnlstm_24 (CuDNNLSTM)    (None, 64)                49664
_________________________________________________________________
dropout_36 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 65
=================================================================
Total params: 2,836,417
Trainable params: 2,836,353
Non-trainable params: 64
_________________________________________________________________
None
```

In [49]:

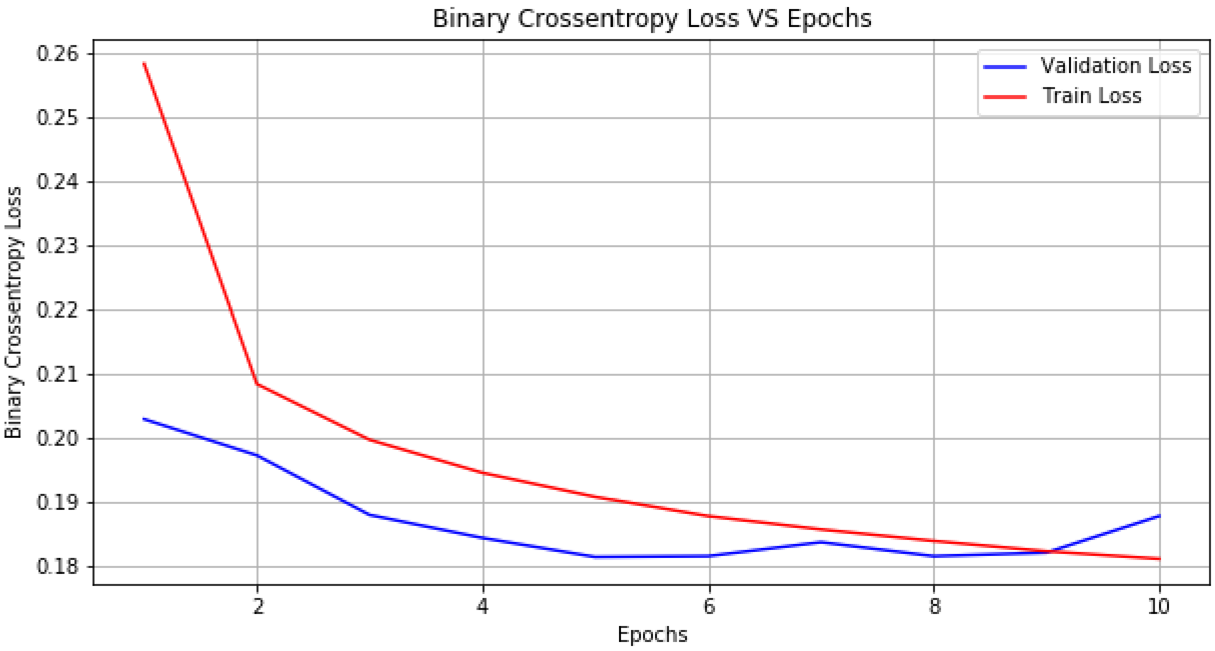
```
# Fitting the data to the model
history3 = model_3.fit(x_train, y_train, nb_epoch=10, batch_size=256 ,verbose=1,validation_data=(x_test, y_test)
```

```
c: 0.92 - ETA: 32s - loss: 0.1813 - acc: 0.92 - ETA: 32s - loss: 0.1813 - acc: 0.92 - ETA: 31s - loss: 0.181
3 - acc: 0.92 - ETA: 30s - loss: 0.1813 - acc: 0.92 - ETA: 30s - loss: 0.1812 - acc: 0.92 - ETA: 29s - loss:
0.1813 - acc: 0.92 - ETA: 28s - loss: 0.1812 - acc: 0.92 - ETA: 28s - loss: 0.1812 - acc: 0.92 - ETA: 27s -
loss: 0.1812 - acc: 0.92 - ETA: 27s - loss: 0.1812 - acc: 0.92 - ETA: 26s - loss: 0.1812 - acc: 0.92 - ETA:
25s - loss: 0.1812 - acc: 0.92 - ETA: 25s - loss: 0.1811 - acc: 0.92 - ETA: 24s - loss: 0.1811 - acc: 0.92 -
ETA: 23s - loss: 0.1811 - acc: 0.92 - ETA: 23s - loss: 0.1812 - acc: 0.92 - ETA: 22s - loss: 0.1812 - acc:
0.92 - ETA: 21s - loss: 0.1812 - acc: 0.92 - ETA: 21s - loss: 0.1811 - acc: 0.92 - ETA: 20s - loss: 0.1811 -
acc: 0.92 - ETA: 19s - loss: 0.1811 - acc: 0.92 - ETA: 19s - loss: 0.1812 - acc: 0.92 - ETA: 18s - loss: 0.1
811 - acc: 0.92 - ETA: 17s - loss: 0.1811 - acc: 0.92 - ETA: 17s - loss: 0.1812 - acc: 0.92 - ETA: 16s - los
s: 0.1812 - acc: 0.92 - ETA: 16s - loss: 0.1812 - acc: 0.92 - ETA: 15s - loss: 0.1812 - acc: 0.92 - ETA: 14s
- loss: 0.1812 - acc: 0.92 - ETA: 14s - loss: 0.1812 - acc: 0.92 - ETA: 13s - loss: 0.1813 - acc: 0.92 - ET
A: 12s - loss: 0.1814 - acc: 0.92 - ETA: 12s - loss: 0.1814 - acc: 0.92 - ETA: 11s - loss: 0.1813 - acc: 0.9
2 - ETA: 10s - loss: 0.1813 - acc: 0.92 - ETA: 10s - loss: 0.1813 - acc: 0.92 - ETA: 9s - loss: 0.1812 - ac
c: 0.9277 - ETA: 8s - loss: 0.1812 - acc: 0.927 - ETA: 8s - loss: 0.1812 - acc: 0.927 - ETA: 7s - loss: 0.18
13 - acc: 0.927 - ETA: 6s - loss: 0.1813 - acc: 0.927 - ETA: 6s - loss: 0.1813 - acc: 0.927 - ETA: 5s - los
s: 0.1812 - acc: 0.927 - ETA: 5s - loss: 0.1812 - acc: 0.927 - ETA: 4s - loss: 0.1812 - acc: 0.927 - ETA: 3s
- loss: 0.1812 - acc: 0.927 - ETA: 3s - loss: 0.1812 - acc: 0.927 - ETA: 2s - loss: 0.1812 - acc: 0.927 - ET
A: 1s - loss: 0.1812 - acc: 0.927 - ETA: 1s - loss: 0.1812 - acc: 0.927 - ETA: 0s - loss: 0.1811 - acc: 0.92
7 - 735s 3ms/step - loss: 0.1812 - acc: 0.9277 - val_loss: 0.1878 - val_acc: 0.9258
```

In [50]:
```
x = list(range(1,11))

# Validation loss
vy = history3.history['val_loss']
# Training loss
ty = history3.history['loss']

# Calling the function to draw the plot
plt_dynamic(x, vy, ty)
```



In [6]:

```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names = ["LSTM layer","Train accuracy","Test accuracy"]
x.add_row(["1x1",0.9286,0.9266])
x.add_row(["2x2",0.9286,0.9266])
x.add_row(["3x3",0.9277,0.9258])

print(x)
```

```
+------------+----------------+---------------+
| LSTM layer | Train accuracy | Test accuracy |
+------------+----------------+---------------+
|    1x1     |     0.9286     |     0.9266    |
|    2x2     |     0.9286     |     0.9266    |
|    3x3     |     0.9277     |     0.9258    |
+------------+----------------+---------------+
```