In [1]:

```
# Credits: https://machinelearningmastery.com/sequence-classification-lstm-recurr
# LSTM for sequence classification in the Amazon dataset
import numpy
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
# fix random seed for reproducibility
numpy.random.seed(7)
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
# Tutorial about Python regular expressions: https://pymotw.com/2/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
```

```
Using TensorFlow backend.
```

In [2]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_i
d=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redi
rect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.go
ogleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdri
ve%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonly%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_type=code (http
s://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pf
ee6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%
3A2.0%3Aoob&scope=email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%2
0https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapi
s.com%2Fauth%2Fdrive.photos.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%
2Fpeopleapi.readonly&response_type=code)

Enter your authorization code:
··········
Mounted at /content/gdrive
```

In [0]:
```
con = sqlite3.connect('/content/gdrive/My Drive/Colab Notebooks/Amazon_LSTM/datab
```

In [11]:
```
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3""",

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

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

```
Number of data points in our data (525814, 10)
```

Out[11]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulness
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1

In [0]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [17]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[17]:

	UserId	ProductId	ProfileName	Time	Score	Text	COUN
0	#oc-R115TNMSPFT9I7	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B007Y59HVM	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ET0	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [18]:
```
# Removing duplicate reviews
final=filtered_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}
print(final.shape)
```

```
(364173, 10)
```

In [19]:
```
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[19]:
```
69.25890143662969
```

In [0]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [21]:
```
#Before starting the next phase of preprocessing lets see the number of entries l
print(final.shape)

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

```
(364171, 10)
```

Out[21]:
```
1    307061
0     57110
Name: Score, dtype: int64
```

In [0]:
```
final["cleanReview"] = final["Summary"].map(str) + ". " + final["Text"]
```

In [23]:
```
final['cleanReview'].head()
```

Out[23]:
```
0    Good Quality Dog Food. I have bought several o...
1    Not as Advertised. Product arrived labeled as ...
2    "Delight" says it all. This is a confection th...
3    Cough Medicine. If you are looking for the sec...
4    Great taffy. Great taffy at a great price.  Th...
Name: cleanReview, dtype: object
```

In [24]:
```
final['lengthOfReview'] = final['cleanReview'].str.split().str.len()
final['lengthOfReview'].head()
```

Out[24]:
```
0    52
1    34
2    98
3    43
4    29
Name: lengthOfReview, dtype: int64
```

In [25]:
```
#remove urls from text python
from tqdm import tqdm
lst = []
removed_urls_list = []
for text in tqdm(final['cleanReview']):
  removed_urls_text = re.sub(r"http\S+", "", text)
  lst.append(removed_urls_text)
```

```
100%|██████████| 364171/364171 [00:00<00:00, 466502.05it/s]
```

In [26]:
```
#remove urls from text python
removed_urls_list = []
for text in tqdm(lst):
  removed_urls_text = re.sub(r"http\S+", "", text)
  removed_urls_list.append(removed_urls_text)
```

```
100%|██████████| 364171/364171 [00:00<00:00, 478211.95it/s]
```

In [27]:

```
from bs4 import BeautifulSoup
text_lst = []
for text in tqdm(removed_urls_list):
  soup = BeautifulSoup(text, 'lxml')
  text = soup.get_text()
  text_lst.append(text)
# print(text)
# print("="*50)
```

```
100%|██████████| 364171/364171 [01:48<00:00, 3342.84it/s]
```

In [28]:

```
print(len(final['cleanReview']))
```

```
364171
```

In [0]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(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
```

In [30]:

```
decat_lst = []
for decat_text in tqdm(text_lst):
  text = decontracted(decat_text)
  decat_lst.append(text)
```

```
100%|██████████| 364171/364171 [00:05<00:00, 69570.14it/s]
```

In [31]:

```
strip_list = []
for to_strip in tqdm(decat_lst):
  text = re.sub("\S*\d\S*", "", to_strip).strip()
  strip_list.append(text)
```

```
100%|██████████| 364171/364171 [00:15<00:00, 22976.41it/s]
```

In [32]:

```

spatial_list = []
for to_spatial in tqdm(strip_list):
  text = re.sub('[^A-Za-z0-9]+', ' ', to_spatial)
  spatial_list.append(text)
```

```
100%|██████████| 364171/364171 [00:09<00:00, 38863.52it/s]
```

In [0]:

```
stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'our
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', '
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itsel
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because'
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'th
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all'
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than',
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "di
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn'
            'won', "won't", 'wouldn', "wouldn't"])
```

In [34]:

```
# Combining all the above stundents
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(spatial_list):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = 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.split() if e.lower() not in s
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 364171/364171 [02:33<00:00, 2373.35it/s]
```

In [35]:

```
print(len(preprocessed_reviews))
preprocessed_reviews[-1]
```

```
364171
```

Out[35]:

```
'great honey satisfied product advertised use cereal raw vinegar general sweetn
er'
```

In [0]:

```
final['cleanReview'] = preprocessed_reviews
```

In [37]:
```
print(len(final))
final.tail(5)
```

```
364171
```

Out[37]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	He
525809	568450	B001EO7N10	A28KG5XORO54AY	Lettie D. Carter	0	0
525810	568451	B003S1WTCU	A3I8AFVPEE8KI5	R. Sawyer	0	0
525811	568452	B004I613EE	A121AA1GQV751Z	pksd "pk_007"	2	2
525812	568453	B004I613EE	A3IBEVCTXKNOH	Kathy A. Welch "katwel"	1	1
525813	568454	B001LR2CU2	A3LGQPJCZVL9UC	srfell17	0	0

In [38]:
```
final['cleanReview'][0]
```

Out[38]:
```
'good quality dog food bought several vitality canned dog food products found g
ood quality product looks like stew processed meat smells better labrador finic
ky appreciates product better'
```

In [39]:
```
final['lengthOfReview'][0]
```

Out[39]:
```
52
```

In [2]:
```
dir_path = os.getcwd()
# conn = sqlite3.connect(os.path.join(dir_path, '/content/gdrive/My Drive/Colab N
conn = sqlite3.connect(os.path.join(dir_path, 'final.sqlite'))
# final.to_sql('Reviews', conn, if_exists='replace', index=False)
```

In [3]:

```
review_3 = pd.read_sql_query(""" SELECT count(*) FROM Reviews""", conn)
print(review_3)
```

```
   count(*)
0    364171
```

In [4]:

```
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews""", conn)
```

In [5]:

```
filtered_data.shape
```

Out[5]:

```
(364171, 12)
```

In [6]:

```
filtered_data["Time"] = pd.to_datetime(filtered_data["Time"], unit = "s")
filtered_data = filtered_data.sort_values(by = "Time")
```

In [7]:

```

filtered_data.head(5)
```

Out[7]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Hel
117924	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0
117901	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2
298792	451856	B00004CXX9	AIUWLEQ1ADEG5	Elizabeth Medina	0	0
169281	230285	B00004RYGX	A344SMIA5JECGM	Vincent P. Ross	1	2
298791	451855	B00004CXX9	AJH6LUC1UT1ON	The Phantom of the Opera	0	0

In [8]:

```
print(len(filtered_data))
filtered_data.info()
filtered_data = filtered_data
print(len(filtered_data))
```

```
364171
<class 'pandas.core.frame.DataFrame'>
Int64Index: 364171 entries, 117924 to 107253
Data columns (total 12 columns):
Id                          364171 non-null int64
ProductId                   364171 non-null object
UserId                      364171 non-null object
ProfileName                 364171 non-null object
HelpfulnessNumerator        364171 non-null int64
HelpfulnessDenominator      364171 non-null int64
Score                       364171 non-null int64
Time                        364171 non-null datetime64[ns]
Summary                     364171 non-null object
Text                        364171 non-null object
cleanReview                 364171 non-null object
lengthOfReview              364171 non-null int64
dtypes: datetime64[ns](1), int64(5), object(6)
memory usage: 36.1+ MB
364171
```

In [9]:

```
filtered_data['Score'].value_counts()
```

Out[9]:

```
1    307061
0     57110
Name: Score, dtype: int64
```

In [10]:

```
len(filtered_data['lengthOfReview'])
```

Out[10]:

```
364171
```

In [11]:

```
X = filtered_data["cleanReview"]
print(print("shape of X:", X.head(5)))
y = filtered_data["Score"]
print("shape of y:", y.head(5))
X_len = filtered_data['lengthOfReview']
```

```
shape of X: 117924    every book educational witty little book makes...
117901    whole series great way spend time child rememb...
298792    entertainingl funny beetlejuice well written m...
169281    modern day fairy tale twist rumplestiskin capt...
298791    fantastic beetlejuice excellent funny movie ke...
Name: cleanReview, dtype: object
None
shape of y: 117924    1
117901    1
298792    1
169281    1
298791    1
Name: Score, dtype: int64
```

Vocabulary of all the words

In [12]:

```
# all_data = pd.read_sql_query(""" SELECT * FROM Reviews""", con)
```

In [13]:

```
complete_review = filtered_data["cleanReview"]
```

In [14]:

```
complete_review.head(5)
```

Out[14]:

```
117924    every book educational witty little book makes...
117901    whole series great way spend time child rememb...
298792    entertainingl funny beetlejuice well written m...
169281    modern day fairy tale twist rumplestiskin capt...
298791    fantastic beetlejuice excellent funny movie ke...
Name: cleanReview, dtype: object
```

In [15]:

```
complete_review[0]
```

Out[15]:

```
'good quality dog food bought several vitality canned dog food products found g
ood quality product looks like stew processed meat smells better labrador finic
ky appreciates product better'
```

In [15]:

```
vocab_list = complete_review.astype(str).values.tolist()
```

In [16]:

```
vocab_list_strings = []
for words in vocab_list:
  vocab_list_strings.append(words.split())
```

In [17]:

```
word_list = [word for line in vocab_list for word in line.split()]
```

In [18]:

```
word_list[0]
```

Out[18]:

```
'every'
```

In [19]:

```
from collections import Counter
counts = list(Counter(word_list).items())
```

In [20]:

```
print(counts[0])
print(len(counts))
```

```
('kapooey', 1)
122244
```

In [21]:

```
# counts[:100]
sorted_freq = sorted(counts, key=lambda x: x[1], reverse = True)
```

```
In [22]: sorted_freq[:10]
```

```
Out[22]: [('not', 402056),
          ('like', 161348),
          ('great', 156532),
          ('good', 156003),
          ('taste', 115761),
          ('product', 111441),
          ('one', 110340),
          ('coffee', 101230),
          ('tea', 98173),
          ('love', 96957)]
```

```
In [23]: top_words = sorted_freq[:5000]
```

```
In [24]: vocab_list_strings[0]
```

```
Out[24]: ['every',
          'book',
          'educational',
          'witty',
          'little',
          'book',
          'makes',
          'son',
          'laugh',
          'loud',
          'recite',
          'car',
          'driving',
          'along',
          'always',
          'sing',
          'refrain',
          'learned',
          'whales',
          'india',
          'drooping',
          'roses',
          'love',
          'new',
          'words',
          'book',
          'introduces',
          'silliness',
          'classic',
          'book',
          'willing',
          'bet',
          'son',
          'still',
          'able',
          'recite',
          'memory',
          'college']
```

In [25]:
```
for key, value in top_words[:10]:
  print(key, value)
```

```
not 402056
like 161348
great 156532
good 156003
taste 115761
product 111441
one 110340
coffee 101230
tea 98173
love 96957
```

In [26]:
```
from tqdm import tqdm
all_list = []
for ind_line in tqdm(vocab_list_strings):
    in_num = []
    for single_review in ind_line:
        for key, value in top_words:
            if key == single_review:
                new_review = value
                break
            else:
                new_review = 0
        in_num.append(new_review)
    all_list.append(in_num)
```

```
100%|██████████| 364171/364171 [40:05<00:00, 151.38it/s]
```

In [28]:
```
import pickle
file = open('all_list', 'wb')
pickle.dump(all_list,file)
file.close()
```

In [29]:
```
print(all_list[0])
print(len(all_list))
```

```
[24310, 1608, 0, 0, 53562, 1608, 25031, 7873, 0, 283, 0, 1841, 458, 5336, 2373
6, 0, 0, 1826, 0, 907, 0, 403, 96957, 17728, 1346, 1608, 0, 0, 1858, 1608, 114
2, 828, 7873, 27303, 11340, 0, 487, 1107]
364171
```

In [30]:
```
print(len(y))
```

```
364171
```

In [20]:
```
X_train = all_list[0:80000]
Y_train = y[0:80000]
X_test = all_list[80000:100000]
Y_test = y[80000:100000]
```

In [23]:
```
print(X_train_padded.shape)
print(len(Y_train))
print(X_test_padded.shape)
print(len(Y_test))
```

```
(80000, 600)
80000
(20000, 600)
20000
```

In [22]:
```
max_review_length = 600
X_train_padded = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test_padded = sequence.pad_sequences(X_test, maxlen=max_review_length)
```

In [108]:
```
print(X_train_padded.shape)
print(X_train_padded[1])
```

```
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0 402056
   2195 111441  12294    941    285   1146   3424   3424  16594  20873
   2731    438 402056  19596    551   1888    899      0 111441    285]
```

In [109]:

```
# create the model
top_words = 5000
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(500000, embedding_vecor_length, input_length=max_review_lengt
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_10 (Embedding)     (None, 600, 32)           16000000  
_________________________________________________________________
lstm_10 (LSTM)               (None, 100)               53200     
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 101       
=================================================================
Total params: 16,053,301
Trainable params: 16,053,301
Non-trainable params: 0
_________________________________________________________________
None
```

In [111]:

```
print(X_train_padded.shape)
print(len(Y_train))
print(top_words)
print(embedding_vecor_length)
print(max_review_length)
```

```
(80000, 600)
80000
5000
32
600
```

In [112]:

```
print(X_train_padded.shape)
print(len(Y_train))
print(X_test_padded.shape)
print(len(Y_test))
```

```
(80000, 600)
80000
(20000, 600)
20000
```

In [ ]:

```
history = model.fit(X_train_padded, Y_train, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test_padded, Y_test, verbose=0)
```

```
/home/j_choudhary1001/anaconda3/lib/python3.5/site-packages/ipykernel/__main__.
py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  if __name__ == '__main__':
```

```
Epoch 1/10
80000/80000 [==============================] - 819s 10ms/step - loss: 0.2275 -
acc: 0.9119
Epoch 2/10
80000/80000 [==============================] - 820s 10ms/step - loss: 0.1594 -
acc: 0.9387
Epoch 3/10
80000/80000 [==============================] - 829s 10ms/step - loss: 0.1448 -
acc: 0.9441
Epoch 4/10
80000/80000 [==============================] - 822s 10ms/step - loss: 0.1328 -
acc: 0.9490
Epoch 5/10
80000/80000 [==============================] - 810s 10ms/step - loss: 0.1203 -
acc: 0.9539
Epoch 6/10
80000/80000 [==============================] - 807s 10ms/step - loss: 0.1086 -
acc: 0.9594
Epoch 7/10
80000/80000 [==============================] - 806s 10ms/step - loss: 0.0975 -
acc: 0.9632
Epoch 8/10
80000/80000 [==============================] - 802s 10ms/step - loss: 0.0890 -
acc: 0.9676
Epoch 9/10
80000/80000 [==============================] - 812s 10ms/step - loss: 0.0803 -
acc: 0.9705
Epoch 10/10
80000/80000 [==============================] - 814s 10ms/step - loss: 0.0727 -
acc: 0.9738
```

In [114]:

```
print("Accuracy: %.2f%%" % (scores[1]*100))
```

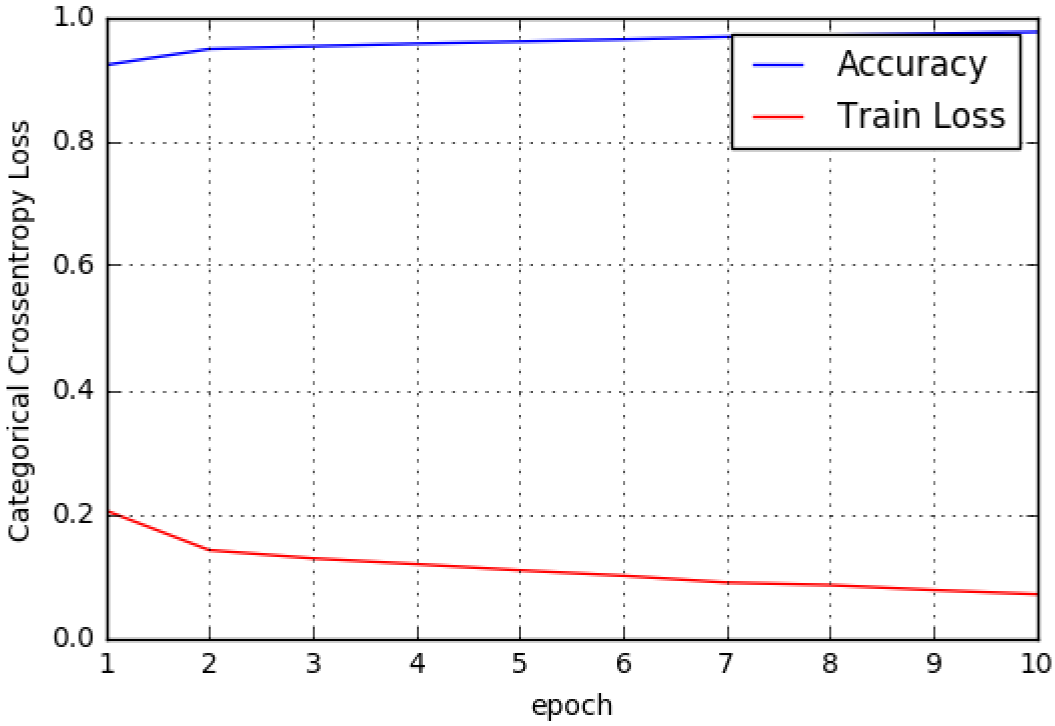
```
Accuracy: 94.27%
```

In [116]:

```
import pickle
file = open('1_layer_model', 'wb')
pickle.dump(model,file)
file.close()
```

In [42]:

```python
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['acc']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Accuracy")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



2 LSTM layers

In [35]:

```
# create the model
top_words = 5000
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(500000, embedding_vecor_length, input_length=max_review_lengt
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.5/site-pack
ages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from ten
sorflow.python.framework.ops) is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 600, 32)           16000000  
_________________________________________________________________
lstm_1 (LSTM)                (None, 600, 100)          53200     
_________________________________________________________________
lstm_2 (LSTM)                (None, 100)               80400     
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101       
=================================================================
Total params: 16,133,701
Trainable params: 16,133,701
Non-trainable params: 0
_________________________________________________________________
None
```

In [36]:

```
history = model.fit(X_train_padded, Y_train, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test_padded, Y_test, verbose=0)
```

```
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.5/site-pack
ages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.o
ps.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.

/home/j_choudhary1001/anaconda3/lib/python3.5/site-packages/ipykernel/__main__.
py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  if __name__ == '__main__':

Epoch 1/10
80000/80000 [==============================] - 1572s 20ms/step - loss: 0.2062 -
acc: 0.9246
Epoch 2/10
80000/80000 [==============================] - 1566s 20ms/step - loss: 0.1453 -
acc: 0.9437
Epoch 3/10
80000/80000 [==============================] - 1552s 19ms/step - loss: 0.1297 -
acc: 0.9514
Epoch 4/10
80000/80000 [==============================] - 1551s 19ms/step - loss: 0.1156 -
acc: 0.9572
Epoch 5/10
80000/80000 [==============================] - 1553s 19ms/step - loss: 0.1060 -
acc: 0.9611
Epoch 6/10
80000/80000 [==============================] - 1551s 19ms/step - loss: 0.0944 -
acc: 0.9663
Epoch 7/10
80000/80000 [==============================] - 1548s 19ms/step - loss: 0.0844 -
acc: 0.9700
Epoch 8/10
80000/80000 [==============================] - 1549s 19ms/step - loss: 0.0764 -
acc: 0.9732
Epoch 9/10
80000/80000 [==============================] - 1550s 19ms/step - loss: 0.0696 -
acc: 0.9760
Epoch 10/10
80000/80000 [==============================] - 1549s 19ms/step - loss: 0.0620 -
acc: 0.9793
```

In [37]:

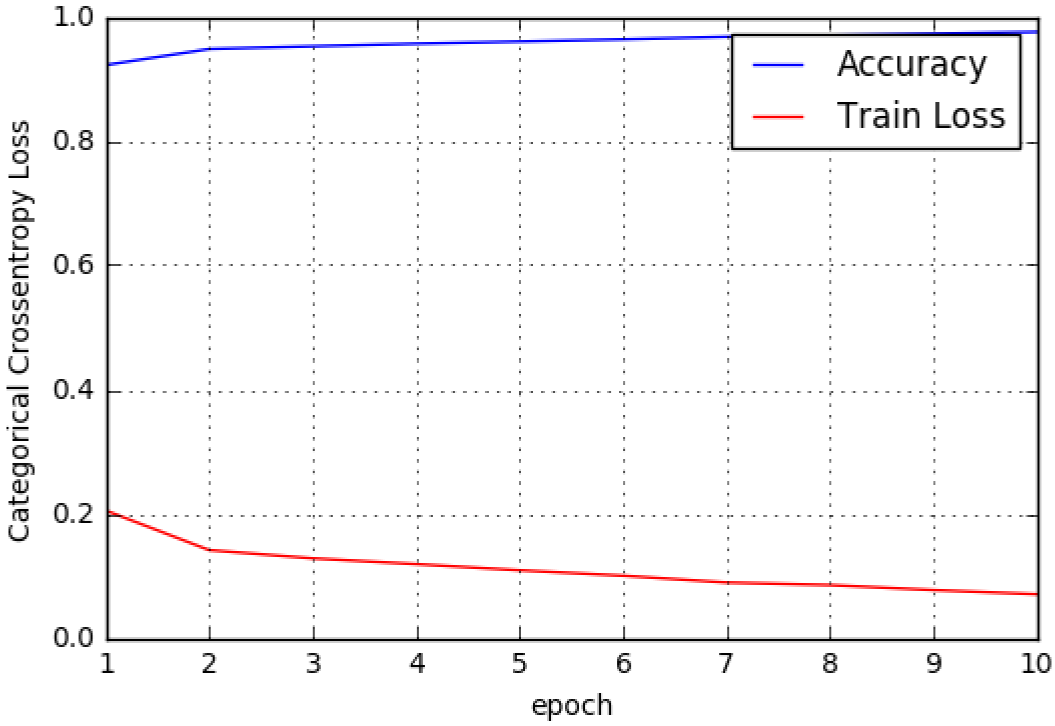
```
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 93.86%
```

In [38]:

```
import pickle
file = open('2_layer_model', 'wb')
pickle.dump(model,file)
file.close()
```

In [41]:
```
import matplotlib.pyplot as plt
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['acc']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Accuracy")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```



In [18]:
```
import pickle
with open(r"all_list", "rb") as input_file:
    all_list = pickle.load(input_file)
```

In [19]:
```
len(all_list)
```

Out[19]:
```
364171
```

In [24]:

```
from keras.layers import Dropout
# create the model
top_words = 5000
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(500000, embedding_vecor_length, input_length=max_review_lengt
model.add(LSTM(100))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

```
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.5/site-pack
ages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from ten
sorflow.python.framework.ops) is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.5/site-pack
ages/keras/backend/tensorflow_backend.py:3445: calling dropout (from tensorflo
w.python.ops.nn_ops) with keep_prob is deprecated and will be removed in a futu
re version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - kee
p_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_1 (Embedding)      (None, 600, 32)           16000000  
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200     
_________________________________________________________________
dropout_1 (Dropout)          (None, 100)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101       
=================================================================
Total params: 16,053,301
Trainable params: 16,053,301
Non-trainable params: 0
_________________________________________________________________
None
```

In [25]:

```
print(X_train_padded.shape)
print(len(Y_train))
print(top_words)
print(embedding_vecor_length)
print(max_review_length)
```

```
(80000, 600)
80000
5000
32
600
```

In [112]:
```
print(X_train_padded.shape)
print(len(Y_train))
print(X_test_padded.shape)
print(len(Y_test))
```

```
(80000, 600)
80000
(20000, 600)
20000
```

In [26]:
```
history = model.fit(X_train_padded, Y_train, nb_epoch=10, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test_padded, Y_test, verbose=0)
```

```
WARNING:tensorflow:From /home/j_choudhary1001/anaconda3/lib/python3.5/site-pack
ages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.o
ps.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
```

```
/home/j_choudhary1001/anaconda3/lib/python3.5/site-packages/ipykernel/__main__.
py:1: UserWarning: The `nb_epoch` argument in `fit` has been renamed `epochs`.
  if __name__ == '__main__':
```

```
Epoch 1/10
80000/80000 [==============================] - 844s 11ms/step - loss: 0.2056 -
acc: 0.9225
Epoch 2/10
80000/80000 [==============================] - 835s 10ms/step - loss: 0.1417 -
acc: 0.9484
Epoch 3/10
80000/80000 [==============================] - 836s 10ms/step - loss: 0.1284 -
acc: 0.9528
Epoch 4/10
80000/80000 [==============================] - 837s 10ms/step - loss: 0.1191 -
acc: 0.9569
Epoch 5/10
80000/80000 [==============================] - 826s 10ms/step - loss: 0.1092 -
acc: 0.9604
Epoch 6/10
80000/80000 [==============================] - 824s 10ms/step - loss: 0.1005 -
acc: 0.9639
Epoch 7/10
80000/80000 [==============================] - 823s 10ms/step - loss: 0.0894 -
acc: 0.9681
Epoch 8/10
80000/80000 [==============================] - 813s 10ms/step - loss: 0.0854 -
acc: 0.9702
Epoch 9/10
80000/80000 [==============================] - 811s 10ms/step - loss: 0.0773 -
acc: 0.9730
Epoch 10/10
80000/80000 [==============================] - 815s 10ms/step - loss: 0.0706 -
acc: 0.9759
```

In [28]:

```
print("Accuracy: %.2f%%" % (scores[1]*100))
```

```
Accuracy: 94.06%
```

In [35]:

```
history.history
```

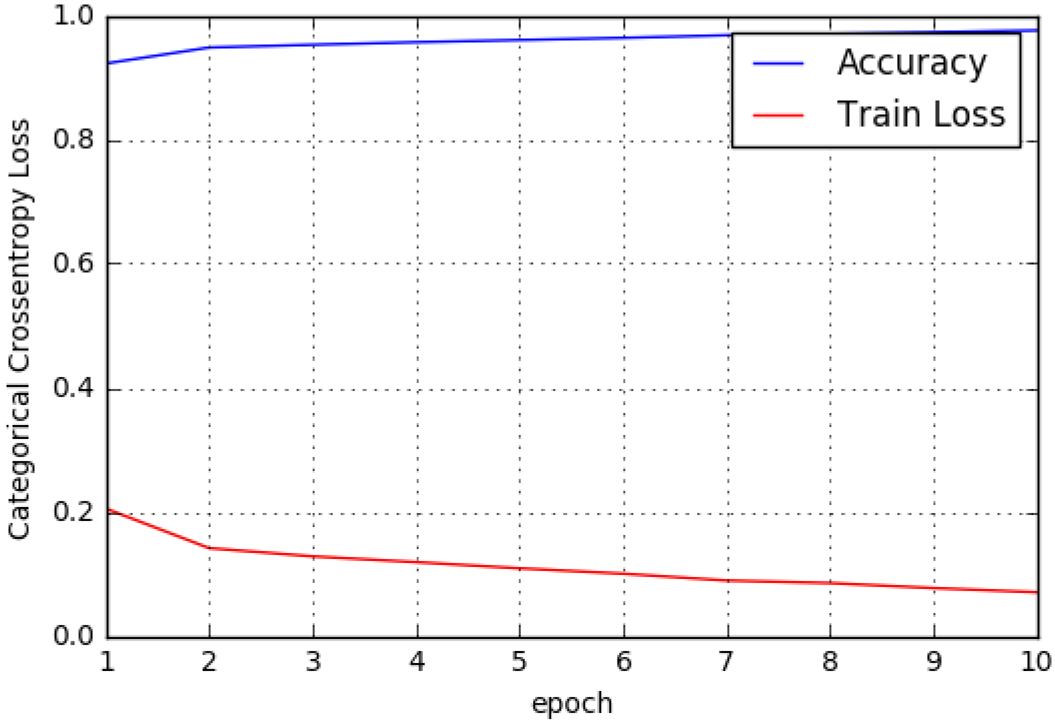
Out[35]:

```
{'acc': [0.922525,
  0.9483875,
  0.9528125,
  0.9569,
  0.9603875,
  0.9639125,
  0.968125,
  0.9702125,
  0.972975,
  0.9758875],
 'loss': [0.20557125323414802,
  0.14170007075816393,
  0.12843381003141402,
  0.11912987211495638,
  0.10917609816044568,
  0.10047360937222838,
  0.08944849496409298,
  0.08537244392558932,
  0.07729678378850222,
  0.07055910568237304]}
```

In [39]:

```
import matplotlib.pyplot as plt
%matplotlib inline
epochs = 10
score = model.evaluate(X_test_padded, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch') ; ax.set_ylabel('Categorical Crossentropy Loss')
x = list(range(1,epochs+1))
vy = history.history['acc']
ty = history.history['loss']
# plt_dynamic(x, vy, ty, ax)
ax.plot(x, vy, 'b', label="Accuracy")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
plt.grid()
fig.canvas.draw()
```

```
Test score: 0.1654715100876987
Test accuracy: 0.94055
```



In [43]:

```
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Number of LSTM layers", "Accuracy"]

x.add_row(["1 layer", "94.27%"])
x.add_row(["2 layer", "93.86%"])
x.add_row(["1 layer with dropout rate = 0.5", "94.06%"])
print(x)
```

```
+---------------------------------+----------+
|      Number of LSTM layers      | Accuracy |
+---------------------------------+----------+
|             1 layer             |  94.27%  |
|             2 layer             |  93.86%  |
| 1 layer with dropout rate = 0.5 |  94.06%  |
+---------------------------------+----------+
```

Procedure followed

1. Preprocessed the Amazon fine food dataset
2. Created the vocabulary
3. Indexed the words based on frequency
4. Made the dataset same as that with IMdb dataset
5. Trained the LSTM model with 1 layer
6. Trained the LSTM model with 2 layers