LSTM Network on Amazon food reviews data

1. Libraries used

In [1]:

```
import sqlite3

import pandas as pd
import numpy as np
from collections import Counter

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve, roc_auc_score
from sklearn.calibration import calibration_curve

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.metrics import binary_crossentropy
from keras.activations import sigmoid,tanh
from keras.optimizers import adam
from keras import layers
from keras.initializers import glorot_normal

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

```
Using TensorFlow backend.
```

2. Loading the data

In [2]:

```
conn = sqlite3.connect(database=r'data/database.sqlite')
data = pd.read_sql_query(sql="select combined_text_clean, score_encoded from ReviewsAf
neg_class,pos_class = dict(Counter(data['score_encoded'])).get(0),dict(Counter(data['s
diff = pos_class - neg_class
upsample_set = data.query('score_encoded == 0').sample(n=diff,replace=True)
data = pd.concat(objs=[data,upsample_set],axis=0)
dict(Counter(data['score_encoded']))
mean_review_length = int(round(np.mean(data['combined_text_clean'].apply(lambda x:len(
```

3. Train, CV and test set splits

In [3]:

```python
X_train,X_test = train_test_split(data,test_size=0.2,stratify=data['score_encoded'])
X_train,X_cv = train_test_split(X_train,test_size=0.2,stratify=X_train['score_encoded'
Y_train = X_train['score_encoded']
Y_cv = X_cv['score_encoded']
Y_test = X_test['score_encoded']
X_train.drop(labels=['score_encoded'],axis=1,inplace=True)
X_cv.drop(labels=['score_encoded'],axis=1,inplace=True)
X_test.drop(labels=['score_encoded'],axis=1,inplace=True)

```

```
d:\byron\documents\projects\2_appliedai\rnn-keras\env\lib\site-packages\pand
as\core\frame.py:3940: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  errors=errors)
```

4. Preprocessing of data

4.1. Get vocab and frequency

In [4]:

```python
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(X_train['combined_text_clean'])
vocab = vectorizer.get_feature_names()
freq = np.sum(X.toarray(),axis=0)
word_freq = pd.DataFrame(list(zip(vocab,freq)),columns=['word','freq'])
word_freq.sort_values(by='freq',ascending=False,inplace=True)
word_freq.reset_index(drop=True, inplace=True)

```

4.2. Store word, index and frequencies in hash tables

In [5]:

```
word_freq_hash = dict()
word_index_hash = dict()
word_index_hash_top = dict()
iteration=1
topwords = 500
for i in word_freq.iterrows():
    word_index_hash[i[1][0]] = i[0]
    word_freq_hash[i[1][0]] = i[1][1]
    if iteration <= topwords:
        word_index_hash_top[i[1][0]] = i[0]
    iteration += 1

def string_encode(string):
    list_of_words = string.split()
    list_of_index = list()
    for word in list_of_words:
        if word_index_hash_top.get(word,0) != 0:
            list_of_index.append(word_index_hash_top.get(word,0))
        else:
            continue
    return list_of_index

```

4.3. Transform reviews into a padded sequence of indexes

In [6]:

```
sequence_length = mean_review_length
X_train = sequence.pad_sequences(sequences=list(X_train['combined_text_clean'].apply(s
X_cv = sequence.pad_sequences(sequences=list(X_cv['combined_text_clean'].apply(string_
X_test = sequence.pad_sequences(sequences=list(X_test['combined_text_clean'].apply(str

```

5. Build LSTM models

In [7]:

```
model1 = Sequential()
model1.add(layers.Embedding(input_dim=len(vocab), output_dim=60, input_length=sequence_
model1.add(layers.Dropout(rate=0.5))
model1.add(layers.LSTM(units=100, activation=tanh, dropout=0.4, recurrent_dropout=0.4,
model1.add(layers.Dropout(rate=0.5))
model1.add(layers.Dense(units=1, activation=sigmoid, kernel_initializer=glorot_normal(

model2 = Sequential()
model2.add(layers.Embedding(input_dim=len(vocab), output_dim=60, input_length=sequence_
model2.add(layers.Dropout(rate=0.2))
model2.add(layers.LSTM(units=100, activation=tanh, dropout=0.3, recurrent_dropout=0.3,
model2.add(layers.LSTM(units=100, activation=tanh, dropout=0.3, recurrent_dropout=0.3,
model2.add(layers.Dropout(rate=0.2))
model2.add(layers.Dense(units=1, activation=sigmoid, kernel_initializer=glorot_normal(

model3 = Sequential()
model3.add(layers.Embedding(input_dim=len(vocab), output_dim=60, input_length=sequence_
model3.add(layers.Dropout(rate=0.2))
model3.add(layers.LSTM(units=100, activation=tanh, dropout=0.3, recurrent_dropout=0.3,
model3.add(layers.Dropout(rate=0.2))
model3.add(layers.Dense(units=10, activation=sigmoid, kernel_initializer=glorot_normal
model3.add(layers.Dropout(rate=0.6))
model3.add(layers.Dense(units=1, activation=sigmoid, kernel_initializer=glorot_normal(

```

```
WARNING: Logging before flag parsing goes to stderr.
W0703 07:00:59.518229 18728 deprecation_wrapper.py:119] From d:\byron\docume
nts\projects\2_appliedai\rnn-keras\env\lib\site-packages\keras\backend\tenso
rflow_backend.py:74: The name tf.get_default_graph is deprecated. Please use
tf.compat.v1.get_default_graph instead.

W0703 07:01:00.718382 18728 deprecation_wrapper.py:119] From d:\byron\docume
nts\projects\2_appliedai\rnn-keras\env\lib\site-packages\keras\backend\tenso
rflow_backend.py:517: The name tf.placeholder is deprecated. Please use tf.c
ompat.v1.placeholder instead.

W0703 07:01:00.927820 18728 deprecation_wrapper.py:119] From d:\byron\docume
nts\projects\2_appliedai\rnn-keras\env\lib\site-packages\keras\backend\tenso
rflow_backend.py:4138: The name tf.random_uniform is deprecated. Please use
tf.random.uniform instead.

W0703 07:01:01.039522 18728 deprecation_wrapper.py:119] From d:\byron\docume
nts\projects\2_appliedai\rnn-keras\env\lib\site-packages\keras\backend\tenso
rflow_backend.py:133: The name tf.placeholder_with_default is deprecated. Pl
ease use tf.compat.v1.placeholder_with_default instead.

W0703 07:01:01.049495 18728 deprecation.py:506] From d:\byron\documents\proj
ects\2_appliedai\rnn-keras\env\lib\site-packages\keras\backend\tensorflow_ba
ckend.py:3445: calling dropout (from tensorflow.python.ops.nn_ops) with keep
_prob is deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
W0703 07:01:01.096129 18728 deprecation_wrapper.py:119] From d:\byron\docume
nts\projects\2_appliedai\rnn-keras\env\lib\site-packages\keras\backend\tenso
rflow_backend.py:4185: The name tf.truncated_normal is deprecated. Please us
e tf.random.truncated_normal instead.
```

```
W0703 07:01:02.649525 18728 nn_ops.py:4224] Large dropout rate: 0.6 (>0.5).
In TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Please
ensure that this is intended.
```

6. Training the different models

In [8]:

```
def training_models(model,modelname,X_train,Y_train,X_cv,Y_cv,X_test,Y_test,epochs=5,b

    results = dict()

    print('BUILDING MODEL '+ modelname)

    model.summary()

    model.compile(optimizer=adam(),loss=binary_crossentropy,metrics=['accuracy'])

    print('Training the model...')

    history = model.fit(x=X_train,y=Y_train,epochs=epochs, batch_size=batchsize,valida

    print('Storing the training and CV loss...')

    results[modelname] = {'loss':history.history['loss'], 'val_loss':history.history['

    print('Storing the model accuracy...')

    accuracy_train,accuracy_cv = model.evaluate(X_train, Y_train, verbose=0),model.eva

    pred = model.predict_classes(x=X_test)
    pred_score = model.predict(x=X_test)

    results.get(modelname)['accuracy on train'] = round(accuracy_train[1]*100,2)
    results.get(modelname)['accuracy on cv'] = round(accuracy_cv[1]*100,2)
    results.get(modelname)['accuracy on test'] = round(accuracy_score(y_true=Y_test,y_

    print('Storing fpr, tpr and thresholds...')

    fpr,tpr,thresholds = roc_curve(y_true=Y_test,y_score=pred_score)
    results.get(modelname)['fpr'] = fpr
    results.get(modelname)['tpr'] = tpr
    results.get(modelname)['thresholds'] = thresholds

    print('Storing auc values...')

    auc = roc_auc_score(y_true=Y_test,y_score=pred_score)

    results.get(modelname)['auc'] = round(auc*100,2)

    print('Storing true and predicted prob...')

    prob_true,prob_pred = calibration_curve(y_true=Y_test,y_prob=pred_score)

    results.get(modelname)['true_prob'] = prob_true
    results.get(modelname)['pred_prob'] = prob_pred

    for key,value in results.items():

        con_matrix = confusion_matrix(y_true=Y_test,y_pred=pred)
        sns.heatmap(data=con_matrix,annot=True,fmt="d")
        plt.title(modelname + ' Confusion Matrix')
        plt.show();

        plt.plot(value['loss'],color='darkorange',label=' Training Loss')
        plt.plot(value['val_loss'],color='red',label=' CV Loss')
        plt.xlabel('epoch')
        plt.ylabel('binary log loss')
        plt.title(key + ' Training curve')
        plt.legend()
        plt.show();

        plt.plot([0, 1], [0, 1], color='darkorange', linestyle='--',label='random mode
        plt.plot(value['fpr'],value['tpr'],color='red',label='ROC curve (area = %0.2f)
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title(key + ' Receiver operating characteristic')
        plt.legend(loc="lower right")
        plt.show();

        plt.plot([0, 1], [0, 1], color='darkorange', linestyle='--')
        plt.plot(value['true_prob'],value['pred_prob'],color='red')
        plt.xlabel('True prob')
        plt.ylabel('Pred prob')
        plt.title(key + ' Calibration Curve')
        plt.legend(loc="lower right")
        plt.show();

    return results

```

In [9]:

```
model1_results = training_models(model=model1,modelname='model1',X_train=X_train,Y_tra
                                X_cv=X_cv,Y_cv=Y_cv,X_test=X_test,Y_test=Y_test,epochs
```

```
W0703 07:01:02.834344 18728 deprecation_wrapper.py:119] From d:\byron\docu
ments\projects\2_appliedai\rnn-keras\env\lib\site-packages\keras\optimizer
s.py:790: The name tf.train.Optimizer is deprecated. Please use tf.compat.
v1.train.Optimizer instead.

W0703 07:01:02.848306 18728 deprecation.py:323] From d:\byron\documents\pr
ojects\2_appliedai\rnn-keras\env\lib\site-packages\tensorflow\python\ops\n
n_impl.py:180: add_dispatch_support.<locals>.wrapper (from tensorflow.pyth
on.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
```

In [10]:

```
model2_results = training_models(model=model2,modelname='model2',X_train=X_train,Y_tra
                                X_cv=X_cv,Y_cv=Y_cv,X_test=X_test,Y_test=Y_test,epochs
```

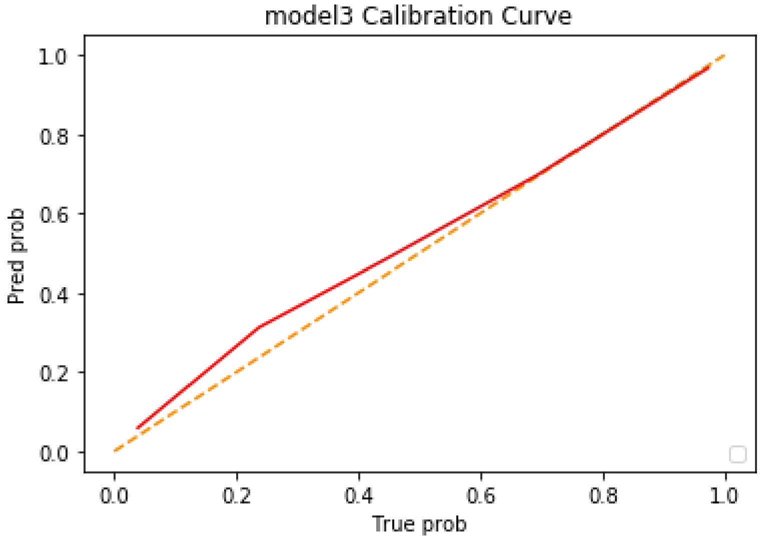
```
BUILDING MODEL model2
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
embedding_2 (Embedding)      (None, 206, 60)           2141460   
_________________________________________________________________
dropout_3 (Dropout)          (None, 206, 60)           0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 206, 100)          64400     
_________________________________________________________________
lstm_3 (LSTM)                (None, 100)               80400     
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101       
=================================================================
Total params: 2,286,361
Trainable params: 2,286,361
Non-trainable params: 0
```

In [11]:

```
model3_results = training_models(model=model3,modelname='model3',X_train=X_train,Y_tra
                                X_cv=X_cv,Y_cv=Y_cv,X_test=X_test,Y_test=Y_test,epochs

```

```
W0703 09:06:07.763583 18728 legend.py:1282] No handles with labels found t
o put in legend.
```

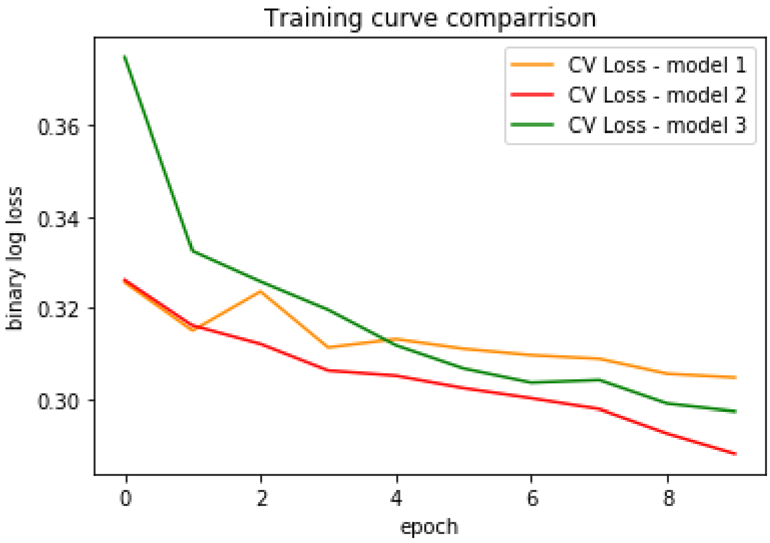


7. Putting the model results together

In [12]:

```
plt.plot(model1_results['model1']['val_loss'],color='darkorange',label='CV Loss - mode
plt.plot(model2_results['model2']['val_loss'],color='red',label='CV Loss - model 2')
plt.plot(model3_results['model3']['val_loss'],color='green',label='CV Loss - model 3')
plt.xlabel('epoch')
plt.ylabel('binary log loss')
plt.title('Training curve comparrison')
plt.legend()
plt.show();

i=1
model_compare = dict()
for model in (model1_results,model2_results,model3_results):
    modeltext = 'model'+str(i)
    avg_train_loss = np.mean(model[modeltext]['loss'])
    avg_val_loss = np.mean(model[modeltext]['val_loss'])
    acc_on_train = model[modeltext]['accuracy on train']
    acc_on_cv = model[modeltext]['accuracy on cv']
    acc_on_test = model[modeltext]['accuracy on test']
    auc = model[modeltext]['auc']
    model_compare[modeltext] = {'avg_train_loss':avg_train_loss,'avg_val_loss':avg_val_
                                'acc_on_cv':acc_on_cv,'acc_on_test':acc_on_test,'auc':
    i+=1
pd.DataFrame(model_compare)
```



Out[12]:

	model1	model2	model3
acc_on_cv	86.770000	87.74000	87.050000
acc_on_test	87.170000	87.97000	87.290000
acc_on_train	87.890000	89.23000	87.940000
auc	94.570000	95.13000	94.780000
avg_train_loss	0.332330	0.31060	0.389516
avg_val_loss	0.312889	0.30471	0.317586

8. Using the model on a random query point

In [13]:

```python
X_q = 'I am happy with the service'
X_q = sequence.pad_sequences(sequences=list(pd.Series(X_q).apply(string_encode).values
output = model2.predict_classes(x=X_q)[0]
if output == 0:
    print('negative review')
else:
    print('positive review')

X_q = 'I think your service is poor'
X_q = sequence.pad_sequences(sequences=list(pd.Series(X_q).apply(string_encode).values
output = model2.predict_classes(x=X_q)[0]
if output == 0:
    print('negative review')
else:
    print('positive review')

```

```
negative review
negative review
```