Libraries

In [1]:

```
#Data:
import pandas as pd
import numpy as np
from collections import Counter

# Text preprocessing:
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer, one_hot
from sklearn.feature_extraction.text import CountVectorizer

# Layers:
from keras.layers import Input, Embedding, LSTM, Dense, Flatten, concatenate, Dropout,

# Model:
from keras.models import Model

# Metrics:
from sklearn.metrics import roc_auc_score

from time import time
import keras
import matplotlib.pyplot as plt
from keras.callbacks import TensorBoard

from sklearn.utils import class_weight

#%matplotlibinline
```

```
Using TensorFlow backend.
```

Load data

In [2]:

```
data = pd.read_csv('data/final_features.csv')

data.sort_values(by='project_submitted_datetime',inplace=True)

Counter(data['project_is_approved'])
print('ratio excepted: ', round(Counter(data['project_is_approved']).get(1)/data.shape
print('ratio rejected: ', round(Counter(data['project_is_approved']).get(0)/data.shape

```

```
ratio excepted:  85.0 %
ratio rejected:  15.0 %
```

In [3]:

```
for feature in data.iteritems():
    print(feature[0],':','has', str(data[data[feature[0]].isnull().values][feature[0]]
```

```
project_submitted_datetime : has 0 missing values
clean_teacher_prefix : has 0 missing values
clean_school_state : has 0 missing values
clean_grade_categories : has 0 missing values
clean_subject_categories : has 0 missing values
clean_subject_subcategories : has 0 missing values
clean_project_title : has 43 missing values
clean_essay : has 0 missing values
clean_resource_summary : has 0 missing values
resource_summary_contains_numerical_digits : has 0 missing values
std_price : has 0 missing values
std_quantity : has 0 missing values
std_teacher_number_of_previously_posted_projects : has 0 missing values
nrm_price : has 0 missing values
nrm_quantity : has 0 missing values
nrm_teacher_number_of_previously_posted_projects : has 0 missing values
project_is_approved : has 0 missing values
```

In [4]:

```
data.fillna(value={'clean_project_title':''}, inplace=True)
```

In [5]:

```
for feature in data.iteritems():
    print(feature[0],':','has', str(data[data[feature[0]].isnull().values][feature[0]]
```

```
project_submitted_datetime : has 0 missing values
clean_teacher_prefix : has 0 missing values
clean_school_state : has 0 missing values
clean_grade_categories : has 0 missing values
clean_subject_categories : has 0 missing values
clean_subject_subcategories : has 0 missing values
clean_project_title : has 0 missing values
clean_essay : has 0 missing values
clean_resource_summary : has 0 missing values
resource_summary_contains_numerical_digits : has 0 missing values
std_price : has 0 missing values
std_quantity : has 0 missing values
std_teacher_number_of_previously_posted_projects : has 0 missing values
nrm_price : has 0 missing values
nrm_quantity : has 0 missing values
nrm_teacher_number_of_previously_posted_projects : has 0 missing values
project_is_approved : has 0 missing values
```

In [6]:

```
data['total_text_data'] = data['clean_essay']

data.drop(labels=['clean_project_title','clean_essay','clean_resource_summary'],axis=1
```

Split data into train,CV and test

In [7]:

```
# data = data.iloc[0:1000,:]

data_train = data.iloc[0:int(data.shape[0]*0.8),:]
data_train = data_train.iloc[0:int(data_train.shape[0]*0.8),:]
data_cv = data_train.iloc[int(data_train.shape[0]*0.8):,:]
data_test = data.iloc[int(data.shape[0]*0.8):,:]

Y_train = data_train['project_is_approved']
data_train.drop(labels=['project_is_approved'],axis=1,inplace=True)
X_train = data_train

Y_cv = data_cv['project_is_approved']
data_cv.drop(labels=['project_is_approved'],axis=1,inplace=True)
X_cv = data_cv

Y_test = data_test['project_is_approved']
data_test.drop(labels=['project_is_approved'],axis=1,inplace=True)
X_test = data_test

```

```
c:\users\byron\applications\pythonmaster\lib\site-packages\pandas\core\fram
e.py:3697: 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)
```

In [8]:

```
print('X_train shape: ',X_train.shape, 'Y_train shape: ',Y_train.shape)
print('X_cv shape: ',X_cv.shape, 'Y_cv shape: ',Y_cv.shape)
print('X_test shape: ',X_test.shape, 'Y_test shape: ',Y_test.shape)

```

```
X_train shape:  (69918, 14) Y_train shape:  (69918,)
X_cv shape:  (13984, 14) Y_cv shape:  (13984,)
X_test shape:  (21850, 14) Y_test shape:  (21850,)
```

Turn text data into sequence data - text preprocessing

In [9]:

```
# Use training data:
text = Tokenizer()
text.fit_on_texts(X_train['total_text_data'])
text_sequences_train = text.texts_to_sequences(X_train['total_text_data'])

def max_length(observation_text):
    observation_text_lengths = list()
    for obs in observation_text:
        observation_text_lengths.append(len(obs.split()))
    return np.mean(observation_text_lengths)

max_len = int(max_length(X_train['total_text_data'].values))

text_sequences_train = pad_sequences(sequences=text_sequences_train, maxlen=max_len, pa

dictionary = text.word_index

frequencies = text.word_counts
frequencies = dict(frequencies)
vocab_size = len(dictionary.keys()) + 1

# Transform cv and test data
text_sequences_cv = text.texts_to_sequences(X_cv['total_text_data'])
text_sequences_cv = pad_sequences(sequences=text_sequences_cv, maxlen=max_len, padding=
text_sequences_test = text.texts_to_sequences(X_test['total_text_data'])
text_sequences_test = pad_sequences(sequences=text_sequences_test, maxlen=max_len, pad

```

Get pretrained word embeddings

In [10]:

```
embeddings_index = dict()
with open('glove/glove.6B.100d.txt','r',encoding="utf-8") as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.array(values[1:], dtype='float32')
        embeddings_index[word] = coefs
print('Loaded {} word vectors.'.format(len(embeddings_index)))

embedding_matrix = np.zeros((vocab_size, 100))
for word, i in dictionary.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

```

```
Loaded 400000 word vectors.
```

One hot encode categorical features

In [11]:

```
# Encode teacher prefix:
teacher_vec = CountVectorizer(binary=True)
clean_teacher_prefix_ohe_train = teacher_vec.fit_transform(X_train['clean_teacher_prefi
clean_teacher_prefix_ohe_cv = teacher_vec.transform(X_cv['clean_teacher_prefix'])
clean_teacher_prefix_ohe_test = teacher_vec.transform(X_test['clean_teacher_prefix'])

# Encode school state:
school_vec = CountVectorizer(binary=True)
clean_school_state_ohe_train = school_vec.fit_transform(X_train['clean_school_state'])
clean_school_state_ohe_cv = school_vec.transform(X_cv['clean_school_state'])
clean_school_state_ohe_test = school_vec.transform(X_test['clean_school_state'])

# Encode grade categories:
grade_vec = CountVectorizer(binary=True)
clean_grade_categories_ohe_train = grade_vec.fit_transform(X_train['clean_grade_categor
clean_grade_categories_ohe_cv = grade_vec.transform(X_cv['clean_grade_categories'])
clean_grade_categories_ohe_test = grade_vec.transform(X_test['clean_grade_categories']

# Encode subject categories:
cat_vec = CountVectorizer(binary=True)
clean_subject_categories_ohe_train = cat_vec.fit_transform(X_train['clean_subject_categ
clean_subject_categories_ohe_cv = cat_vec.transform(X_cv['clean_subject_categories'])
clean_subject_categories_ohe_test = cat_vec.transform(X_test['clean_subject_categories

# Encode subject subcategories:
subcat_vec = CountVectorizer(binary=True)
clean_subject_subcategories_ohe_train = subcat_vec.fit_transform(X_train['clean_subject
clean_subject_subcategories_ohe_cv = subcat_vec.transform(X_cv['clean_subject_subcateg
clean_subject_subcategories_ohe_test = subcat_vec.transform(X_test['clean_subject_subc
```

In [12]:

```
def concat_numeric_features(dataset):
    if dataset == 'train':
        a = pd.DataFrame(clean_teacher_prefix_ohe_train.toarray())
        b = pd.DataFrame(clean_school_state_ohe_train.toarray())
        c = pd.DataFrame(clean_grade_categories_ohe_train.toarray())
        d = pd.DataFrame(clean_subject_categories_ohe_train.toarray())
        e = pd.DataFrame(clean_subject_subcategories_ohe_train.toarray())
        f = pd.DataFrame(X_train['resource_summary_contains_numerical_digits'].values)
        g = pd.DataFrame(X_train['nrm_teacher_number_of_previously_posted_projects'].va
        h = pd.DataFrame(X_train['nrm_price'].values)
        i = pd.DataFrame(X_train['nrm_quantity'].values)
    elif dataset == 'cv':
        a = pd.DataFrame(clean_teacher_prefix_ohe_cv.toarray())
        b = pd.DataFrame(clean_school_state_ohe_cv.toarray())
        c = pd.DataFrame(clean_grade_categories_ohe_cv.toarray())
        d = pd.DataFrame(clean_subject_categories_ohe_cv.toarray())
        e = pd.DataFrame(clean_subject_subcategories_ohe_cv.toarray())
        f = pd.DataFrame(X_cv['resource_summary_contains_numerical_digits'].values)
        g = pd.DataFrame(X_cv['nrm_teacher_number_of_previously_posted_projects'].valu
        h = pd.DataFrame(X_cv['nrm_price'].values)
        i = pd.DataFrame(X_cv['nrm_quantity'].values)

    else:
        a = pd.DataFrame(clean_teacher_prefix_ohe_test.toarray())
        b = pd.DataFrame(clean_school_state_ohe_test.toarray())
        c = pd.DataFrame(clean_grade_categories_ohe_test.toarray())
        d = pd.DataFrame(clean_subject_categories_ohe_test.toarray())
        e = pd.DataFrame(clean_subject_subcategories_ohe_test.toarray())
        f = pd.DataFrame(X_test['resource_summary_contains_numerical_digits'].values)
        g = pd.DataFrame(X_test['nrm_teacher_number_of_previously_posted_projects'].va
        h = pd.DataFrame(X_test['nrm_price'].values)
        i = pd.DataFrame(X_test['nrm_quantity'].values)

    concat = pd.concat(objs=[a,b,c,d,e,f,g,h,i],axis=1)
    del a,b,c,d,e,f,g,h,i
    return concat

numeric_train = concat_numeric_features(dataset='train')
numeric_cv = concat_numeric_features(dataset='cv')
numeric_test = concat_numeric_features(dataset='test')

numeric_dim = numeric_train.shape[1]

numeric_train = np.array(numeric_train).reshape((numeric_train.shape[0],numeric_train.
numeric_cv = np.array(numeric_cv).reshape((numeric_cv.shape[0],numeric_cv.shape[1],1))
numeric_test = np.array(numeric_test).reshape((numeric_test.shape[0],numeric_test.shap
```

Building the Neural Network

In [13]:

```
text_input = Input(shape=(max_len,),dtype='int32',name='text_input')
embedding = Embedding(input_dim=vocab_size, output_dim=100, weights=[embedding_matrix]
lstm = LSTM(units=100, activation='relu', kernel_initializer='glorot_normal', bias_ini
flatten_1 = Flatten()(lstm)
################################################
numeric_input = Input(shape=(numeric_dim,1), dtype='float32', name='numeric_input')
conv1D_1 = Conv1D(filters=30, kernel_size=4, strides=1, padding='valid', activation='r
conv1D_2 = Conv1D(filters=30, kernel_size=4, strides=1, padding='valid', activation='r
flatten_2 = Flatten()(conv1D_2)
################################################
concat = concatenate([flatten_1,flatten_2])
################################################
dense_1 = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_initia
drop_1 = Dropout(rate=0.5)(dense_1)
dense_2 = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_initia
drop_2 = Dropout(rate=0.5)(dense_2)
dense_3 = Dense(units=100,activation='relu',kernel_initializer='he_normal',bias_initia
output = Dense(units=1,activation='sigmoid',name='output')(dense_3)

```

```
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\tensorflow\python\framework\op_def_library.py:263: colocate_with (fro
m tensorflow.python.framework.ops) is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\keras\backend\tensorflow_backend.py:3445: calling dropout (from tenso
rflow.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`.
```

In [14]:

```
model = Model(inputs=[text_input,numeric_input],outputs=[output])
```

In [15]:

```
1 model.summary()
```

```
_________________________________________________________________________________
_____________________
Layer (type)                 Output Shape              Param #     Connected t
o
=================================================================================
=====================
text_input (InputLayer)      (None, 137)               0
_________________________________________________________________________________
_____________________
numeric_input (InputLayer)   (None, 501, 1)            0
_________________________________________________________________________________
_____________________
embedding_1 (Embedding)      (None, 137, 100)          4668500     text_input
[0][0]
_________________________________________________________________________________
_____________________
conv1d_1 (Conv1D)            (None, 498, 30)           150         numeric_inp
ut[0][0]
_________________________________________________________________________________
_____________________
lstm_1 (LSTM)                (None, 137, 100)          80400       embedding_1
[0][0]
_________________________________________________________________________________
_____________________
conv1d_2 (Conv1D)            (None, 495, 30)           3630        conv1d_1[0]
[0]
_________________________________________________________________________________
_____________________
flatten_1 (Flatten)          (None, 13700)             0           lstm_1[0]
[0]
_________________________________________________________________________________
_____________________
flatten_2 (Flatten)          (None, 14850)             0           conv1d_2[0]
[0]
_________________________________________________________________________________
_____________________
concatenate_1 (Concatenate)  (None, 28550)             0           flatten_1
[0][0]
                                                                   flatten_2
[0][0]
_________________________________________________________________________________
_____________________
dense_1 (Dense)              (None, 100)               2855100     concatenate
_1[0][0]
_________________________________________________________________________________
_____________________
dropout_1 (Dropout)          (None, 100)               0           dense_1[0]
[0]
_________________________________________________________________________________
_____________________
dense_2 (Dense)              (None, 100)               10100       dropout_1
[0][0]
_________________________________________________________________________________
_____________________
```

```
dropout_2 (Dropout)             (None, 100)           0           dense_2[0]
[0]
_________________________________________________________________
_____________________
dense_3 (Dense)                 (None, 100)           10100       dropout_2
[0][0]
_________________________________________________________________
_____________________
output (Dense)                  (None, 1)             101         dense_3[0]
[0]
=================================================================
======================
Total params: 7,628,081
Trainable params: 2,959,581
Non-trainable params: 4,668,500
_________________________________________________________________
_____________________
```

In [16]:

```
1 tensorboard = TensorBoard(log_dir='.logs/{}'.format(time()))
```

In [17]:

```
1 model.compile(optimizer=keras.optimizers.Adam(lr=0.0001), loss='binary_crossentropy',
```

In [18]:

```
class roc_callback(keras.callbacks.Callback):
    def __init__(self,training_data,validation_data):
        self.x = training_data[0]
        self.y = training_data[1]
        self.x_val = validation_data[0]
        self.y_val = validation_data[1]

    def on_train_begin(self, logs={}):
        return

    def on_train_end(self, logs={}):
        return

    def on_epoch_begin(self, epoch, logs={}):
        return

    def on_epoch_end(self, epoch, logs={}):
        y_pred = self.model.predict(self.x)
        roc = roc_auc_score(self.y, y_pred)
        y_pred_val = self.model.predict(self.x_val)
        roc_val = roc_auc_score(self.y_val, y_pred_val)
        print('\rroc-auc: %s - roc-auc_val: %s' % (str(round(roc,4)),str(round(roc_val
        return

    def on_batch_begin(self, batch, logs={}):
        return

    def on_batch_end(self, batch, logs={}):
        return

```

In [19]:

```
class_weights = class_weight.compute_class_weight('balanced',
                                                  np.unique(Y_train),
                                                  Y_train)
```

In [20]:

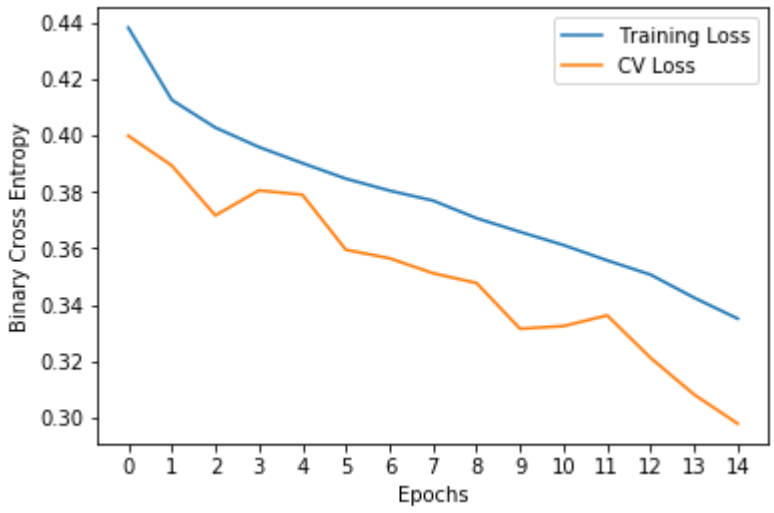
```
batch_size = 100
epochs = 15
history = model.fit(x=[text_sequences_train,numeric_train],y=[Y_train],validation_data=
                        callbacks=[tensorboard,
                        roc_callback(training_data=([text_sequences_train,numeric_trai
                                    validation_data=([text_sequences_cv,numeric_cv], Y
                    class_weight=class_weights
                    )
```

```
WARNING:tensorflow:From c:\users\byron\applications\pythonmaster\lib\site-pa
ckages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.pyt
hon.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Train on 69918 samples, validate on 13984 samples
Epoch 1/15
69918/69918 [==============================] - 159s 2ms/step - loss: 0.4381
- acc: 0.8415 - val_loss: 0.3998 - val_acc: 0.8593
roc-auc: 0.7031 - roc-auc_val: 0.7003
Epoch 2/15
69918/69918 [==============================] - 166s 2ms/step - loss: 0.4125
- acc: 0.8441 - val_loss: 0.3893 - val_acc: 0.8593
roc-auc: 0.7194 - roc-auc_val: 0.7175
Epoch 3/15
69918/69918 [==============================] - 167s 2ms/step - loss: 0.4027
- acc: 0.8440 - val_loss: 0.3716 - val_acc: 0.8594
roc-auc: 0.7322 - roc-auc_val: 0.7298
Epoch 4/15
69918/69918 [==============================] - 165s 2ms/step - loss: 0.3958
- acc: 0.8442 - val_loss: 0.3805 - val_acc: 0.8601
roc-auc: 0.7439 - roc-auc_val: 0.743
Epoch 5/15
69918/69918 [==============================] - 162s 2ms/step - loss: 0.3901
- acc: 0.8449 - val_loss: 0.3789 - val_acc: 0.8666
roc-auc: 0.7514 - roc-auc_val: 0.7512
Epoch 6/15
69918/69918 [==============================] - 163s 2ms/step - loss: 0.3846
- acc: 0.8463 - val_loss: 0.3594 - val_acc: 0.8666
roc-auc: 0.7628 - roc-auc_val: 0.7628
Epoch 7/15
69918/69918 [==============================] - 164s 2ms/step - loss: 0.3804
- acc: 0.8474 - val_loss: 0.3564 - val_acc: 0.8696
roc-auc: 0.7701 - roc-auc_val: 0.7707
Epoch 8/15
69918/69918 [==============================] - 163s 2ms/step - loss: 0.3769
- acc: 0.8489 - val_loss: 0.3511 - val_acc: 0.8703
roc-auc: 0.7796 - roc-auc_val: 0.7814
Epoch 9/15
69918/69918 [==============================] - 165s 2ms/step - loss: 0.3706
- acc: 0.8505 - val_loss: 0.3477 - val_acc: 0.8712
roc-auc: 0.788 - roc-auc_val: 0.7896
Epoch 10/15
69918/69918 [==============================] - 170s 2ms/step - loss: 0.3658
- acc: 0.8521 - val_loss: 0.3315 - val_acc: 0.8734
roc-auc: 0.7984 - roc-auc_val: 0.8017
```

```
Epoch 11/15
69918/69918 [==============================] - 167s 2ms/step - loss: 0.3611
- acc: 0.8550 - val_loss: 0.3324 - val_acc: 0.8770
roc-auc: 0.8089 - roc-auc_val: 0.8117
Epoch 12/15
69918/69918 [==============================] - 169s 2ms/step - loss: 0.3557
- acc: 0.8572 - val_loss: 0.3362 - val_acc: 0.8771
roc-auc: 0.8177 - roc-auc_val: 0.8191
Epoch 13/15
69918/69918 [==============================] - 171s 2ms/step - loss: 0.3506
- acc: 0.8590 - val_loss: 0.3211 - val_acc: 0.8837
roc-auc: 0.8297 - roc-auc_val: 0.8324
Epoch 14/15
69918/69918 [==============================] - 168s 2ms/step - loss: 0.3424
- acc: 0.8630 - val_loss: 0.3082 - val_acc: 0.8854
roc-auc: 0.839 - roc-auc_val: 0.8421
Epoch 15/15
69918/69918 [==============================] - 166s 2ms/step - loss: 0.3351
- acc: 0.8665 - val_loss: 0.2979 - val_acc: 0.8907
roc-auc: 0.8503 - roc-auc_val: 0.852
```

In [21]:

```
plt.plot(np.arange(epochs),history.history['loss'],label='Training Loss')
plt.plot(np.arange(epochs),history.history['val_loss'],label='CV Loss')
plt.xlabel('Epochs')
plt.ylabel('Binary Cross Entropy')
plt.legend()
plt.xticks(np.arange(epochs))
plt.show()
```



In [22]:

```
model.evaluate(x=[text_sequences_test,numeric_test],y=[Y_test],batch_size=batch_size)
```

```
21850/21850 [==============================] - 15s 696us/step
```

Out[22]:

```
[0.3769638879348266, 0.8518077769323127]
```

AUC Score on test data

In [23]:

```
Y_pred = model.predict(x=[text_sequences_test,numeric_test],
                       batch_size=batch_size)
roc_auc_score(Y_test, Y_pred)
```

Out[23]:

```
0.7229880277013565
```