LSTM_Assignment

December 25, 2019

0.1 Assignment : 14

[0]:
```
%tensorflow_version 1.x
```

[2]:
```
import os
import pickle
import numpy as np
import pandas as pd
from math import ceil
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
from prettytable import PrettyTable
from keras.utils import to_categorical
from sklearn.preprocessing import MultiLabelBinarizer, LabelEncoder
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, Input, Dense, Flatten, LSTM,\
                            Dropout, concatenate, Conv1D,BatchNormalization
from tensorflow.keras import regularizers,initializers,optimizers,Model
from keras.callbacks import Callback, ModelCheckpoint
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
Using TensorFlow backend.
```

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

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

[0]:
```
dir_path="/content/drive/My Drive/Colab Notebooks/AppliedAI/LSTM_Donors_Choose/"
```

[4]:
```
table = PrettyTable(field_names=["Model","Weighted Test AUC","Micro Test AUC"])
print(table)
```

```
+-------+-------------------+----------------+
| Model | Weighted Test AUC | Micro Test AUC |
+-------+-------------------+----------------+
+-------+-------------------+----------------+
```

[0]:
```
with open(dir_path+'glove_vectors', 'rb') as f:
    glove = pickle.load(f)
```

[0]:
```
# _df_Resource = pd.read_csv(dir_path+'resources.csv')
# _df_Resource.head()
```

[0]:
```
# _df_train = pd.read_csv(dir_path+'train_data.csv')
# _df_train.head()
```

[7]:
```
preprocessed_df=pd.read_csv(dir_path+'preprocessed_data.csv')
preprocessed_df.head()
```

[7]:
```
  school_state  ...   price
0           ca  ...  725.05
1           ut  ...  213.03
2           ca  ...  329.00
3           ga  ...  481.04
4           wa  ...   17.74

[5 rows x 9 columns]
```

[8]:
```
print(preprocessed_df.shape)
print(preprocessed_df.columns)
```

```
(109248, 9)
Index(['school_state', 'teacher_prefix', 'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price'],
      dtype='object')
```

[0]:
```
project_approved = preprocessed_df.project_is_approved
preprocessed_df.drop(columns='project_is_approved', inplace=True)
```

[0]:
```
# this is random splitting into train and test set
dfX_train, dfX_test, y_train, y_test = train_test_split(preprocessed_df,
                              project_approved,test_size=0.30, random_state = 0,
                              stratify = project_approved)
dfX_cv, dfX_test, y_cv, y_test = train_test_split(dfX_test,
                              y_test,test_size=0.50, random_state = 0,
                              stratify = y_test)
```

[0]:
```
# For column school state

tokenizer_schoolState = Tokenizer(oov_token='<oov>')
tokenizer_schoolState.fit_on_texts(dfX_train.school_state.to_list())
```

[0]:
```
max_schoolState_length = dfX_train.school_state.apply(lambda x : len(x.split('␣
 ↪'))).max()
tokenised_schoolState_train = tokenizer_schoolState.
 ↪texts_to_sequences(dfX_train.school_state)
tokenised_schoolState_test = tokenizer_schoolState.texts_to_sequences(dfX_test.
 ↪school_state)
tokenised_schoolState_cv = tokenizer_schoolState.texts_to_sequences(dfX_cv.
 ↪school_state)
```

[0]:
```
X_train_schoolState = pad_sequences(tokenised_schoolState_train,
                                    maxlen=max_schoolState_length)
X_test_schoolState = pad_sequences(tokenised_schoolState_test,
                                    maxlen=max_schoolState_length)
X_cv_schoolState = pad_sequences(tokenised_schoolState_cv,
                                    maxlen=max_schoolState_length)
```

[0]:
```
# For column teacher_prefix

tokenizer_teacherPrefix = Tokenizer(oov_token='<oov>')
tokenizer_teacherPrefix.fit_on_texts(dfX_train.teacher_prefix.to_list())
```

[0]:
```
max_teacherPrefix_length = dfX_train.teacher_prefix.apply(lambda x : len(x.
 ↪split(' '))).max()
tokenised_teacherPrefix_train = tokenizer_teacherPrefix.
 ↪texts_to_sequences(dfX_train.teacher_prefix)
tokenised_teacherPrefix_test = tokenizer_teacherPrefix.
 ↪texts_to_sequences(dfX_test.teacher_prefix)
tokenised_teacherPrefix_cv = tokenizer_teacherPrefix.texts_to_sequences(dfX_cv.
 ↪teacher_prefix)
```

[0]:
```
X_train_teacherPrefix = pad_sequences(tokenised_teacherPrefix_train,
                                    maxlen=max_teacherPrefix_length)
X_test_teacherPrefix = pad_sequences(tokenised_teacherPrefix_test,
                                    maxlen=max_teacherPrefix_length)
X_cv_teacherPrefix = pad_sequences(tokenised_teacherPrefix_cv,
                                    maxlen=max_teacherPrefix_length)
```

[0]:
```
# For column project_grade_category

tokenizer_pgCategory = Tokenizer(oov_token='<oov>',
                        filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')
tokenizer_pgCategory.fit_on_texts(dfX_train.project_grade_category.to_list())
```

[0]:
```
max_pgCategory_length = dfX_train.project_grade_category.apply(lambda x : len(x.
 ↪split(' '))).max()
```

```
tokenised_pgCategory_train = tokenizer_pgCategory.texts_to_sequences(dfX_train.
 ↪project_grade_category)
tokenised_pgCategory_test = tokenizer_pgCategory.texts_to_sequences(dfX_test.
 ↪project_grade_category)
tokenised_pgCategory_cv = tokenizer_pgCategory.texts_to_sequences(dfX_cv.
 ↪project_grade_category)
```

[0]:
```
X_train_pgCategory = pad_sequences(tokenised_pgCategory_train,
                                    maxlen=max_pgCategory_length)
X_test_pgCategory = pad_sequences(tokenised_pgCategory_test,
                                    maxlen=max_pgCategory_length)
X_cv_pgCategory = pad_sequences(tokenised_pgCategory_cv,
                                    maxlen=max_pgCategory_length)
```

[20]:
```
tokenizer_pgCategory.index_word
```

[20]:
```
{1: '<oov>',
 2: 'grades_prek_2',
 3: 'grades_3_5',
 4: 'grades_6_8',
 5: 'grades_9_12'}
```

[0]:
```
# For column clean_categories

tokenizer_cleanCategory = Tokenizer(oov_token='<oov>',
                          filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')
tokenizer_cleanCategory.fit_on_texts(dfX_train.clean_categories.to_list())
```

[0]:
```
max_cleanCategory_length = dfX_train.clean_categories.apply(lambda x : len(x.
 ↪split(' '))).max()
tokenised_cleanCategory_train = tokenizer_cleanCategory.
 ↪texts_to_sequences(dfX_train.clean_categories)
tokenised_cleanCategory_test = tokenizer_cleanCategory.
 ↪texts_to_sequences(dfX_test.clean_categories)
tokenised_cleanCategory_cv = tokenizer_cleanCategory.texts_to_sequences(dfX_cv.
 ↪clean_categories)
```

[0]:
```
X_train_cleanCategory = pad_sequences(tokenised_cleanCategory_train,
                                    maxlen=max_cleanCategory_length)
X_test_cleanCategory = pad_sequences(tokenised_cleanCategory_test,
                                    maxlen=max_cleanCategory_length)
X_cv_cleanCategory = pad_sequences(tokenised_cleanCategory_cv,
                                    maxlen=max_cleanCategory_length)
```

[0]:
```
# For column clean_subcategories

tokenizer_cleanSubCategory = Tokenizer(oov_token='<oov>',
                          filters='!"#$%&()*+,-./:;<=>?@[\\]^`{|}~\t\n')
tokenizer_cleanSubCategory.fit_on_texts(dfX_train.clean_subcategories.to_list())
```

```
[0]: max_cleanSubCategory_length = dfX_train.clean_subcategories.apply(lambda x :␣
      ↪len(x.split(' '))).max()
     tokenised_cleanSubCategory_train = tokenizer_cleanSubCategory.
      ↪texts_to_sequences(dfX_train.clean_subcategories)
     tokenised_cleanSubCategory_test = tokenizer_cleanSubCategory.
      ↪texts_to_sequences(dfX_test.clean_subcategories)
     tokenised_cleanSubCategory_cv = tokenizer_cleanSubCategory.
      ↪texts_to_sequences(dfX_cv.clean_subcategories)
```

```
[0]: X_train_cleanSubCategory = pad_sequences(tokenised_cleanSubCategory_train,
                                     maxlen=max_cleanSubCategory_length)
     X_test_cleanSubCategory = pad_sequences(tokenised_cleanSubCategory_test,
                                     maxlen=max_cleanSubCategory_length)
     X_cv_cleanSubCategory = pad_sequences(tokenised_cleanSubCategory_cv,
                                     maxlen=max_cleanSubCategory_length)
```

```
[0]: # For column essay

     tokenizer_Essay = Tokenizer(oov_token='<oov>')
     tokenizer_Essay.fit_on_texts(dfX_train.essay.to_list())
```

```
[0]: # we found the max_essay_length using max length of list of the tokens
     max_essay_length = 350
     tokenised_essay_train = tokenizer_Essay.texts_to_sequences(dfX_train.essay)
     tokenised_essay_test = tokenizer_Essay.texts_to_sequences(dfX_test.essay)
     tokenised_essay_cv = tokenizer_Essay.texts_to_sequences(dfX_cv.essay)
```

```
[0]: X_train_essay = pad_sequences(tokenised_essay_train, maxlen=max_essay_length)
     X_test_essay = pad_sequences(tokenised_essay_test, maxlen=max_essay_length)
     X_cv_essay = pad_sequences(tokenised_essay_cv, maxlen=max_essay_length)
```

```
[0]: EMBEDDING_DIMS = 300      # glove vectors are 300 dims
     VOCAB_SIZE = len(list(tokenizer_Essay.word_counts.keys()))
     embedding_matrix = np.zeros((VOCAB_SIZE+1, EMBEDDING_DIMS))
     for word, i in tokenizer_Essay.word_index.items():
         embedding_vector = glove.get(word)
         if embedding_vector is not None:
             # words not found in embedding index will be all-zeros.
             embedding_matrix[i-1] = embedding_vector
```

```
[31]: preprocessed_df.columns
```

```
[31]: Index(['school_state', 'teacher_prefix', 'project_grade_category',
              'teacher_number_of_previously_posted_projects', 'clean_categories',
              'clean_subcategories', 'essay', 'price'],
             dtype='object')
```

0.1.1 Model-1

Build and Train deep neural network as shown below ref: https://i.imgur.com/w395Yk9.png

- **Input_seq_total_text_data** — You have to give Total text data columns. After this use the Embedding layer to get word vectors. Use given predefined glove word vectors, don't train any word vectors. After this use LSTM and get the LSTM output and Flatten that output.
- **Input_school_state** — Give 'school_state' column as input to embedding layer and Train the Keras Embedding layer.
- **Project_grade_category** — Give 'project_grade_category' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_categories** — Give 'input_clean_categories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** — Give 'input_clean_subcategories' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_clean_subcategories** — Give 'input_teacher_prefix' column as input to embedding layer and Train the Keras Embedding layer.
- **Input_remaining_teacher_number_of_previously_posted_projects._resource_summary_contains_nume** —concatenate remaining columns and add a Dense layer after that.

- For LSTM, you can choose your sequence padding methods on your own or you can train your LSTM without padding, there is no restriction on that.

Below is an example of embedding layer for a categorical columns. In below code all are dummy values, we gave only for referance.

[0]:
```
from sklearn.metrics import roc_auc_score,accuracy_score
import tensorflow as tf

# def auc( y_true, y_pred ):
#     # This is for calculating the AUC Score
#     # Added this code as sometimes the y_true comes to be of a single class
#     # causing error
#     # https://stackoverflow.com/questions/45139163/
↪roc-auc-score-only-one-class-present-in-y-true

#     score = 0.0
#     if len(np.unique(y_true)) == 1: # bug in roc_auc_score
#         score = tf.py_func(lambda y_true, y_pred : accuracy_score(y_true,
#                                          y_pred).astype('float32'),
#                           [y_true, y_pred],
#                           'float32',
#                           stateful=True,
#                           name='sklearnAUC' )
#     else:
#         score = tf.py_func( lambda y_true, y_pred : roc_auc_score(y_true,␣
↪y_pred,
#                           average='weighted', sample_weight=None).
↪astype('float32'),
#                           [y_true, y_pred],
#                           'float32',
#                         stateful=True,
#                         name='sklearnAUC' )

#     return score

def auc_temp(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred, average='weighted')

def auc(y_true, y_pred):
    return tf.py_function(auc_temp, (y_true, y_pred), tf.double)
```

[0]:
```
# https://stats.stackexchange.com/questions/270546/
 ↪how-does-keras-embedding-layer-work
# input_layer = Input(shape=(n,))
# embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
# flatten = Flatten()(embedding)
```

[33]:
```
# school state imput
school_state_inp = Input(shape=(max_schoolState_length,), dtype='int32',
                         name='school_state_inp')
embedded_school_state = Embedding(input_dim=len(tokenizer_schoolState.
 ↪word_index.items()),   # 51
                                                                          ␣
 ↪output_dim=6,name='embedded_school_state')(school_state_inp)
school_state_out = Flatten()(embedded_school_state)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/keras/initializers.py:119: calling
RandomUniform.__init__ (from tensorflow.python.ops.init_ops) with dtype is
deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling
BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops)
with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

[0]:
```
# teacher prefix
teacher_Pref_inp = Input(shape=(max_teacherPrefix_length,),dtype='int32',
                         name='teacher_Pref_inp')
embedded_teacher_Pref = Embedding(input_dim=len(tokenizer_teacherPrefix.
 ↪word_index.items()),
                    ␣
↪output_dim=2,name='embedded_teacher_Pref')(teacher_Pref_inp)
teacher_Pref_out = Flatten()(embedded_teacher_Pref)
```

[0]:
```
# project grade category
pgCategory_Inp = Input(shape=(max_pgCategory_length,),dtype='int32',
                       name='pgCategory_Inp')
embedded_pgCategory = Embedding(input_dim=len(tokenizer_pgCategory.word_index.
↪items()),
                        output_dim=2,name='embedded_pgCategory')(pgCategory_Inp)
pgCategory_Out = Flatten()(embedded_pgCategory)
```

[0]:
```
# project clean_categories
cleanCategory_Inp = Input(shape=(max_cleanCategory_length,),dtype='int32',
                          name='cleanCategory_Inp')
embedded_cleanCategory = Embedding(input_dim=len(tokenizer_cleanCategory.
↪word_index.items()),
                    ␣
↪output_dim=3,name='embedded_cleanCategory')(cleanCategory_Inp)
cleanCategory_Out = Flatten()(embedded_cleanCategory)
```

[0]:
```
# project clean_subcategories
clean_subcategories_Inp =␣
↪Input(shape=(max_cleanSubCategory_length,),dtype='int32',
                                name='clean_subcategories_Inp')
embedded_cleanSubCategory = Embedding(input_dim=len(tokenizer_cleanSubCategory.
↪word_index.items()),
                    ␣
↪output_dim=5,name='embedded_cleanSubCategory')(clean_subcategories_Inp)
clean_subcategories_Out = Flatten()(embedded_cleanSubCategory)
```

[0]:
```
# essay
essay_Inp = Input(shape=(max_essay_length,),dtype='int32',
                  name='essay_Inp')
embedded_Essay = Embedding(input_dim=len(tokenizer_Essay.word_index.items()),
                        output_dim=300,name='embedded_Essay',
                        weights=[embedding_matrix],
                        trainable=False)(essay_Inp)
essay_LSTM = LSTM(units=128, return_sequences=True)(embedded_Essay)
essay_Out = Flatten()(essay_LSTM)
```

[39]:
```
# concatenating remaining columns teacher_number_of_previously_posted_projects
# and price

dfX_train['remaining_cols']=dfX_train[['teacher_number_of_previously_posted_projects',
    'price']].apply(
    lambda x : [x.teacher_number_of_previously_posted_projects, x.price],
    axis=1)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
```

[0]:
```
dfX_test['remaining_cols']=dfX_test[['teacher_number_of_previously_posted_projects',
    'price']].apply(
    lambda x : [x.teacher_number_of_previously_posted_projects, x.price],
    axis=1)
```

[0]:
```
dfX_cv['remaining_cols']=dfX_cv[['teacher_number_of_previously_posted_projects',
    'price']].apply(
    lambda x : [x.teacher_number_of_previously_posted_projects, x.price],
    axis=1)
```

[0]:
```
remaining_cols_Inp = Input(shape=(2,), dtype='float32',␣
 ↪name='remaining_cols_Inp')
dense_remaining = Dense(16,activation='relu',
                        kernel_regularizer=regularizers.l2(0.001),
                        kernel_initializer=initializers.he_normal())
remaining_cols_Out = dense_remaining(remaining_cols_Inp)
```

[0]:
```
# concatenating all the outputs

concatenated_Outs = concatenate([school_state_out, teacher_Pref_out,
                pgCategory_Out,cleanCategory_Out,clean_subcategories_Out,
                essay_Out,remaining_cols_Out])
```

[0]:
```
outPut = Dense(128,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(concatenated_Outs)
outPut = Dropout(0.4)(outPut)
outPut = Dense(64,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(outPut)
outPut = Dropout(0.4)(outPut)
outPut = BatchNormalization()(outPut)
outPut = Dense(32,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(outPut)
outPut = Dropout(0.4)(outPut)
outPut = Dense(2, activation = 'softmax')(outPut)
```

[0]:
```
# create model with all the previously defined inputs
model1 = Model([school_state_inp,teacher_Pref_inp,pgCategory_Inp,
                cleanCategory_Inp,clean_subcategories_Inp,essay_Inp,
                 remaining_cols_Inp], outPut)
model1.compile(loss='categorical_crossentropy',
               optimizer=optimizers.Adam(lr=0.0006,decay = 1e-4),
               metrics=[auc])
print(model1.summary())
```

```
Model: "model"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
essay_Inp (InputLayer)          [(None, 350)]        0

________________________________________________________________________________
__________________
school_state_inp (InputLayer)   [(None, 1)]          0

________________________________________________________________________________
__________________
teacher_Pref_inp (InputLayer)   [(None, 1)]          0

________________________________________________________________________________
__________________
pgCategory_Inp (InputLayer)     [(None, 1)]          0

________________________________________________________________________________
__________________
cleanCategory_Inp (InputLayer)  [(None, 3)]          0

________________________________________________________________________________
__________________
clean_subcategories_Inp (InputL [(None, 3)]          0

________________________________________________________________________________
__________________
embedded_Essay (Embedding)      (None, 350, 300)     14799000    essay_Inp[0][0]

________________________________________________________________________________
__________________
embedded_school_state (Embeddin (None, 1, 6)         312
school_state_inp[0][0]

________________________________________________________________________________
__________________
embedded_teacher_Pref (Embeddin (None, 1, 2)         12
teacher_Pref_inp[0][0]

________________________________________________________________________________
__________________
embedded_pgCategory (Embedding) (None, 1, 2)         10
pgCategory_Inp[0][0]

________________________________________________________________________________
__________________
embedded_cleanCategory (Embeddi (None, 3, 3)         30
cleanCategory_Inp[0][0]
```

```
________________________________________________________________________________
__________________
embedded_cleanSubCategory (Embe (None, 3, 5)         155
clean_subcategories_Inp[0][0]

________________________________________________________________________________
__________________
lstm (LSTM)                     (None, 350, 128)     219648
embedded_Essay[0][0]

________________________________________________________________________________
__________________
remaining_cols_Inp (InputLayer) [(None, 2)]          0

________________________________________________________________________________
__________________
flatten (Flatten)               (None, 6)            0
embedded_school_state[0][0]

________________________________________________________________________________
__________________
flatten_1 (Flatten)             (None, 2)            0
embedded_teacher_Pref[0][0]

________________________________________________________________________________
__________________
flatten_2 (Flatten)             (None, 2)            0
embedded_pgCategory[0][0]

________________________________________________________________________________
__________________
flatten_3 (Flatten)             (None, 9)            0
embedded_cleanCategory[0][0]

________________________________________________________________________________
__________________
flatten_4 (Flatten)             (None, 15)           0
embedded_cleanSubCategory[0][0]

________________________________________________________________________________
__________________
flatten_5 (Flatten)             (None, 44800)        0           lstm[0][0]

________________________________________________________________________________
__________________
dense (Dense)                   (None, 16)           48
remaining_cols_Inp[0][0]

________________________________________________________________________________
__________________
concatenate (Concatenate)       (None, 44850)        0           flatten[0][0]
                                                                 flatten_1[0][0]
                                                                 flatten_2[0][0]
                                                                 flatten_3[0][0]
                                                                 flatten_4[0][0]
                                                                 flatten_5[0][0]
                                                                 dense[0][0]

________________________________________________________________________________
```

```
------------------
dense_1 (Dense)                 (None, 128)          5740928
concatenate[0][0]

--------------------------------------------------------------------------------
------------------
dropout (Dropout)               (None, 128)          0           dense_1[0][0]

--------------------------------------------------------------------------------
------------------
dense_2 (Dense)                 (None, 64)           8256        dropout[0][0]

--------------------------------------------------------------------------------
------------------
dropout_1 (Dropout)             (None, 64)           0           dense_2[0][0]

--------------------------------------------------------------------------------
------------------
batch_normalization (BatchNorma (None, 64)           256         dropout_1[0][0]

--------------------------------------------------------------------------------
------------------
dense_3 (Dense)                 (None, 32)           2080
batch_normalization[0][0]

--------------------------------------------------------------------------------
------------------
dropout_2 (Dropout)             (None, 32)           0           dense_3[0][0]

--------------------------------------------------------------------------------
------------------
dense_4 (Dense)                 (None, 2)            66          dropout_2[0][0]
================================================================================
==================
Total params: 20,770,801
Trainable params: 5,971,673
Non-trainable params: 14,799,128

--------------------------------------------------------------------------------
------------------
None
```

[0]:
```
# Lets run the model
filepath=dir_path+"best_weights.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc',
                    verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]

# At first we got error after 14 epochs so to complete the full 20 epochs we
# will load the best_weights and re-run the fit part

if os.path.isfile(dir_path+'best_weights.hdf5'):
  model1.load_weights(dir_path+'best_weights.hdf5')

model1.fit([X_train_schoolState,X_train_teacherPrefix,X_train_pgCategory,
           X_train_cleanCategory,X_train_cleanSubCategory,X_train_essay,
           np.array(dfX_train['remaining_cols'].to_list())],␣
 ↪to_categorical(y_train),
           epochs=7,verbose=2,batch_size=256,validation_split=0.3,
            callbacks =callbacks_list)

# Epoch 00013: val_auc did not improve from 0.76058
# 53531/53531 - 152s - loss: 0.3956 - auc: 0.7626 - val_loss: 0.4010 - val_auc:␣
 ↪0.7597
```

```
WARNING:tensorflow:The `nb_epoch` argument in `fit` has been renamed `epochs`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/math_grad.py:1424: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 53531 samples, validate on 22942 samples
Epoch 1/20

Epoch 00001: val_auc improved from -inf to 0.59106, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 151s - loss: 0.7298 - auc: 0.5241 - val_loss: 0.5772 - val_auc:
0.5911
Epoch 2/20

Epoch 00002: val_auc improved from 0.59106 to 0.60180, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 151s - loss: 0.5841 - auc: 0.5468 - val_loss: 0.5339 - val_auc:
0.6018
Epoch 3/20

Epoch 00003: val_auc did not improve from 0.60180
53531/53531 - 151s - loss: 0.5439 - auc: 0.5456 - val_loss: 0.5224 - val_auc:
0.5959
Epoch 4/20

Epoch 00004: val_auc improved from 0.60180 to 0.61066, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 149s - loss: 0.5168 - auc: 0.5606 - val_loss: 0.5040 - val_auc:
0.6107
Epoch 5/20

Epoch 00005: val_auc did not improve from 0.61066
```

```
53531/53531 - 146s - loss: 0.4972 - auc: 0.5688 - val_loss: 0.4876 - val_auc:
0.6026
Epoch 6/20

Epoch 00006: val_auc did not improve from 0.61066
53531/53531 - 146s - loss: 0.4843 - auc: 0.5767 - val_loss: 0.4798 - val_auc:
0.6013
Epoch 7/20

Epoch 00007: val_auc did not improve from 0.61066
53531/53531 - 150s - loss: 0.4731 - auc: 0.5804 - val_loss: 0.4686 - val_auc:
0.6020
Epoch 8/20

Epoch 00008: val_auc did not improve from 0.61066
53531/53531 - 151s - loss: 0.4643 - auc: 0.5878 - val_loss: 0.4617 - val_auc:
0.6106
Epoch 9/20

Epoch 00009: val_auc improved from 0.61066 to 0.61604, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 150s - loss: 0.4581 - auc: 0.5905 - val_loss: 0.4552 - val_auc:
0.6160
Epoch 10/20

Epoch 00010: val_auc improved from 0.61604 to 0.61677, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 143s - loss: 0.4514 - auc: 0.5979 - val_loss: 0.4534 - val_auc:
0.6168
Epoch 11/20

Epoch 00011: val_auc improved from 0.61677 to 0.61981, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 145s - loss: 0.4447 - auc: 0.6031 - val_loss: 0.4462 - val_auc:
0.6198
Epoch 12/20

Epoch 00012: val_auc did not improve from 0.61981
53531/53531 - 146s - loss: 0.4408 - auc: 0.6093 - val_loss: 0.4451 - val_auc:
0.6171
Epoch 13/20

Epoch 00013: val_auc improved from 0.61981 to 0.65394, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
```

```
53531/53531 - 150s - loss: 0.4335 - auc: 0.6327 - val_loss: 0.4381 - val_auc:
0.6539
Epoch 14/20

Epoch 00014: val_auc improved from 0.65394 to 0.67504, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 153s - loss: 0.4262 - auc: 0.6588 - val_loss: 0.4270 - val_auc:
0.6750
Epoch 15/20

Epoch 00015: val_auc improved from 0.67504 to 0.67685, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 152s - loss: 0.4212 - auc: 0.6657 - val_loss: 0.4228 - val_auc:
0.6769
Epoch 16/20

Epoch 00016: val_auc did not improve from 0.67685
53531/53531 - 149s - loss: 0.4168 - auc: 0.6703 - val_loss: 0.4563 - val_auc:
0.6090
Epoch 17/20

Epoch 00017: val_auc improved from 0.67685 to 0.70445, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 148s - loss: 0.4150 - auc: 0.6716 - val_loss: 0.4179 - val_auc:
0.7044
Epoch 18/20

Epoch 00018: val_auc did not improve from 0.70445
53531/53531 - 148s - loss: 0.4224 - auc: 0.6342 - val_loss: 0.4329 - val_auc:
0.6150
Epoch 19/20

Epoch 00019: val_auc did not improve from 0.70445
53531/53531 - 146s - loss: 0.4211 - auc: 0.6443 - val_loss: 0.4170 - val_auc:
0.7032
Epoch 20/20

Epoch 00020: val_auc improved from 0.70445 to 0.71357, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights.hdf5
53531/53531 - 148s - loss: 0.4076 - auc: 0.6948 - val_loss: 0.4089 - val_auc:
0.7136
```

```
[0]: <tensorflow.python.keras.callbacks.History at 0x7fb45bfdc160>
```

[0]:
```
# AUC for test data
print("Test AUC for Model1 : %0.3f"%roc_auc_score(to_categorical(y_test),
      model1.predict([X_test_schoolState,X_test_teacherPrefix,X_test_pgCategory,
      X_test_cleanCategory,X_test_cleanSubCategory,
      X_test_essay,np.array(dfX_test['remaining_cols'].to_list())]),
      average='weighted'))
```

```
Test AUC for Model1 : 0.762
```

[0]:
```
# AUC for test data
print("Test AUC for Model1 : %0.3f"%roc_auc_score(to_categorical(y_test),
      model1.predict([X_test_schoolState,X_test_teacherPrefix,X_test_pgCategory,
      X_test_cleanCategory,X_test_cleanSubCategory,
      X_test_essay,np.array(dfX_test['remaining_cols'].to_list())]),
      average='micro'))
```

```
Test AUC for Model1 : 0.916
```

[0]:
```
table.add_row(['Model 1',0.762,0.916])
```

0.1.2 1. Go through this blog, if you have any doubt on using predefined Embedding values in Embedding layer - https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

0.1.3 2. Please go through this link https://keras.io/getting-started/functional-api-guide/ and check the 'Multi-input and multi-output models' then you will get to know how to give multiple inputs.

0.1.4 Model-2

Use the same model as above but for 'input_seq_total_text_data' give only some words in the sentance not all the words. Filter the words as below.

[0]:
```
tfIdf_vectorizer = TfidfVectorizer(min_df=3)
tfIdf_vectorizer.fit_transform(dfX_train.essay.to_list())
```

[0]:
```
<76473x24416 sparse matrix of type '<class 'numpy.float64'>'
        with 8239388 stored elements in Compressed Sparse Row format>
```

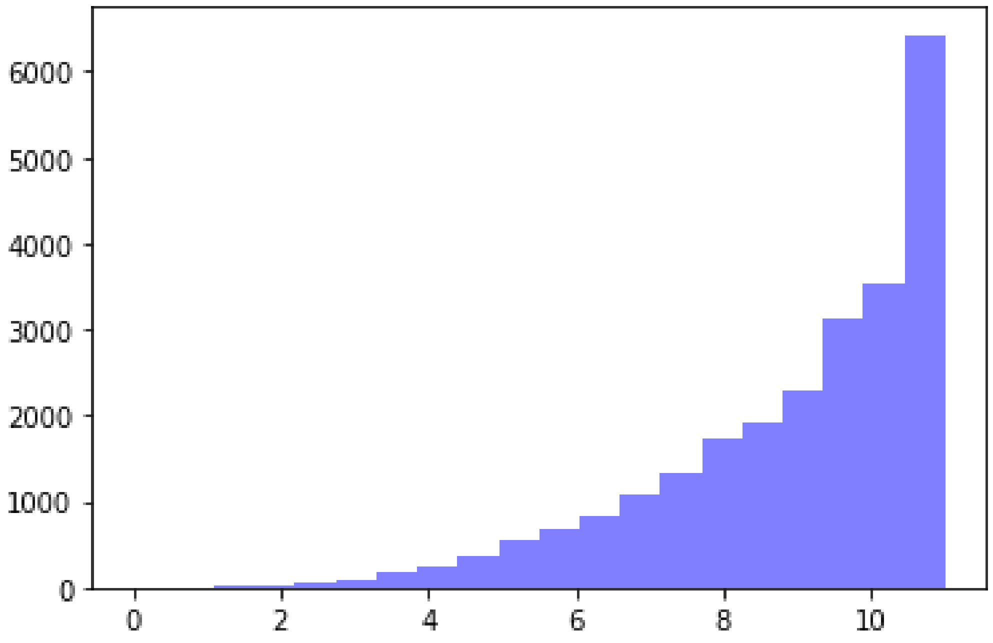
[0]:
```
feature_Idf_Map = dict(zip(tfIdf_vectorizer.
 ↪get_feature_names(),tfIdf_vectorizer.idf_))
```

[0]:
```
len(tfIdf_vectorizer.get_feature_names())
```

[0]:
```
24416
```

[0]:
```
# Histogram plot of the idf values

num_bins = ceil((tfIdf_vectorizer.idf_.max()-tfIdf_vectorizer.idf_.min())/0.5)
hValues, bins, patches = plt.hist(tfIdf_vectorizer.idf_, num_bins,␣
 ↪facecolor='blue',
                alpha=0.5, range=(0, int(tfIdf_vectorizer.idf_.max()+1)))
plt.show()
```



[0]:
```
# Lets study the distribution of words in each of the bins
for bini in range(len(bins)-1):
  print("Bin : ",bini+1)
  print("Range : {} - {}".format(round(bins[bini],3), round(bins[bini+1],3)))
  print("No of words : {} which is {} % of total".format(hValues[bini],
          round((hValues[bini]*100)/len(tfIdf_vectorizer.
 ↪get_feature_names()),2)))
  print("Top 10 words : ")
  words = sorted([(word,score) for word,score in feature_Idf_Map.items()\
                 if score>=bins[bini] and score<=bins[bini+1]],
                 key=lambda x:x[1], reverse=True)[:10]
  print(words)
  print("*"*100)
```

```
Bin :  1
Range : 0.0 - 0.55
No of words : 0.0 which is 0.0 % of total
Top 10 words :
[]
********************************************************************************
********************
Bin :  2
Range : 0.55 - 1.1
```

```
No of words : 2.0 which is 0.01 % of total
Top 10 words :
[('nannan', 1.0460394199858651), ('students', 1.0080085502013645)]
********************************************************************************
********************
Bin :  3
Range : 1.1 - 1.65
No of words : 10.0 which is 0.04 % of total
Top 10 words :
[('many', 1.5766580863856858), ('help', 1.5164292506171393), ('they',
1.5022193040848788), ('the', 1.466500084265604), ('learn', 1.4627750651310945),
('not', 1.4507585683891664), ('classroom', 1.3946702211939477), ('learning',
1.3637010149387487), ('my', 1.244274308661387), ('school', 1.1629790147986256)]
********************************************************************************
********************
Bin :  4
Range : 1.65 - 2.2
No of words : 28.0 which is 0.11 % of total
Top 10 words :
[('provide', 2.191979904762562), ('allow', 2.181014788688464), ('get',
2.1573149022521934), ('every', 2.1551539971560105), ('grade',
2.1145611696147935), ('reading', 2.098037095178083), ('skills',
2.075513707830516), ('want', 2.0679143357229743), ('student',
2.046312622687699), ('time', 2.0446758393448223)]
********************************************************************************
********************
Bin :  5
Range : 2.2 - 2.75
No of words : 63.0 which is 0.26 % of total
Top 10 words :
[('around', 2.7341870570076816), ('us', 2.7271016336092417), ('better',
2.7142406598475963), ('even', 2.7113406360649677), ('level',
2.7050619736461448), ('daily', 2.691627442588886), ('experience',
2.686600349479957), ('become', 2.6858237600802015), ('know', 2.68377927419057),
('used', 2.6820904411947)]
********************************************************************************
********************
Bin :  6
Range : 2.75 - 3.3
No of words : 97.0 which is 0.4 % of total
Top 10 words :
[('still', 3.2945623363056655), ('despite', 3.2926184492165227), ('population',
3.2921007168725738), ('some', 3.2902907616273342), ('groups',
3.282570702698326), ('two', 3.276564678638114), ('ability', 3.2571343423978814),
('everyday', 3.252024341872385), ('outside', 3.251402953671194), ('child',
3.2423743845186848)]
********************************************************************************
********************
```

```
Bin :  7
Range : 3.3 - 3.85
No of words : 187.0 which is 0.77 % of total
Top 10 words :
[('neighborhood', 3.8467469892723885), ('curious', 3.843148244947615),
('creativity', 3.842474919818372), ('setting', 3.840457660364825), ('minds',
3.8379976345239624), ('along', 3.837104577978785), ('programs',
3.8299886929379268), ('grades', 3.829545626913842), ('critical',
3.825787470617031), ('prepare', 3.825125730215691)]
********************************************************************************
********************
Bin :  8
Range : 3.85 - 4.4
No of words : 244.0 which is 1.0 % of total
Top 10 words :
[('highly', 4.3996816755234445), ('regular', 4.3996816755234445),
('alternative', 4.398116117473741), ('expand', 4.398116117473741),
('developing', 4.394602547589346), ('collaborate', 4.393044914875663), ('store',
4.392267007407417), ('achievement', 4.391101279667091), ('passion',
4.391101279667091), ('academics', 4.390713005540683)]
********************************************************************************
********************
Bin :  9
Range : 4.4 - 4.95
No of words : 364.0 which is 1.49 % of total
Top 10 words :
[('excel', 4.94761508760451), ('lost', 4.94761508760451), ('weekly',
4.94626099125678), ('gets', 4.944233278497129), ('drive', 4.942209669037602),
('supports', 4.942209669037602), ('brains', 4.941536041528128), ('mini',
4.9408628674872235), ('increasing', 4.939517877371891), ('beneficial',
4.938174693825423)]
********************************************************************************
********************
Bin :  10
Range : 4.95 - 5.5
No of words : 544.0 which is 2.23 % of total
Top 10 words :
[('chart', 5.4994697432805655), ('known', 5.4994697432805655), ('contribute',
5.498293964191554), ('barriers', 5.497119565935612), ('methods',
5.497119565935612), ('practices', 5.497119565935612), ('aware',
5.495946545273249), ('feet', 5.495946545273249), ('greatest',
5.493604623828167), ('vibrant', 5.493604623828167)]
********************************************************************************
********************
Bin :  11
Range : 5.5 - 6.05
No of words : 665.0 which is 2.72 % of total
Top 10 words :
```

```
[('leading', 6.046227376272619), ('pen', 6.046227376272619), ('saying',
6.046227376272619), ('blue', 6.044196918722237), ('deeply', 6.044196918722237),
('represented', 6.044196918722237), ('strives', 6.044196918722237), ('demand',
6.042170575577005), ('memory', 6.040148330196238), ('sort', 6.040148330196238)]
********************************************************************************
********************
Bin :  12
Range : 6.05 - 6.6
No of words : 817.0 which is 3.35 % of total
Top 10 words :
[('drum', 6.59925919512169), ('hallway', 6.59925919512169), ('heads',
6.59925919512169), ('jumping', 6.59925919512169), ('locations',
6.59925919512169), ('picked', 6.59925919512169), ('sleep', 6.59925919512169),
('tailored', 6.59925919512169), ('cook', 6.595731854603722), ('ended',
6.595731854603722)]
********************************************************************************
********************
Bin :  13
Range : 6.6 - 7.15
No of words : 1065.0 which is 4.36 % of total
Top 10 words :
[('55', 7.144839664940729), ('crisis', 7.144839664940729), ('halls',
7.144839664940729), ('headsets', 7.144839664940729), ('inventions',
7.144839664940729), ('lists', 7.144839664940729), ('moreover',
7.144839664940729), ('observation', 7.144839664940729), ('pivotal',
7.144839664940729), ('rare', 7.144839664940729)]
********************************************************************************
********************
Bin :  14
Range : 7.15 - 7.7
No of words : 1332.0 which is 5.46 % of total
Top 10 words :
[('assistive', 7.690829201164387), ('chore', 7.690829201164387), ('climbing',
7.690829201164387), ('computation', 7.690829201164387), ('contest',
7.690829201164387), ('credits', 7.690829201164387), ('delightful',
7.690829201164387), ('deodorant', 7.690829201164387), ('disc',
7.690829201164387), ('entitled', 7.690829201164387)]
********************************************************************************
********************
Bin :  15
Range : 7.7 - 8.25
No of words : 1736.0 which is 7.11 % of total
Top 10 words :
[('3000', 8.237372907532457), ('affording', 8.237372907532457), ('apt',
8.237372907532457), ('attempts', 8.237372907532457), ('beating',
8.237372907532457), ('boat', 8.237372907532457), ('buzzers', 8.237372907532457),
('captions', 8.237372907532457), ('certificate', 8.237372907532457), ('compact',
8.237372907532457)]
```

```
********************************************************************************
********************
Bin :  16
Range : 8.25 - 8.8
No of words : 1912.0 which is 7.83 % of total
Top 10 words :
[('2003', 8.7789701899652), ('abcs', 8.7789701899652), ('adaptation',
8.7789701899652), ('addiction', 8.7789701899652), ('affiliated',
8.7789701899652), ('astounded', 8.7789701899652), ('audiobook',
8.7789701899652), ('beakers', 8.7789701899652), ('bilingualism',
8.7789701899652), ('bind', 8.7789701899652)]
********************************************************************************
********************
Bin :  17
Range : 8.8 - 9.35
No of words : 2279.0 which is 9.33 % of total
Top 10 words :
[('1990', 9.300267113598487), ('225', 9.300267113598487), ('aac',
9.300267113598487), ('accidental', 9.300267113598487), ('adjusts',
9.300267113598487), ('advantageous', 9.300267113598487), ('affirm',
9.300267113598487), ('alarming', 9.300267113598487), ('amelia',
9.300267113598487), ('ankle', 9.300267113598487)]
********************************************************************************
********************
Bin :  18
Range : 9.35 - 9.9
No of words : 3125.0 which is 12.8 % of total
Top 10 words :
[('165', 9.846810819966556), ('2028', 9.846810819966556), ('215',
9.846810819966556), ('2s', 9.846810819966556), ('340', 9.846810819966556),
('3doodle', 9.846810819966556), ('4cs', 9.846810819966556), ('640',
9.846810819966556), ('90th', 9.846810819966556), ('absorbs', 9.846810819966556)]
********************************************************************************
********************
Bin :  19
Range : 9.9 - 10.45
No of words : 3520.0 which is 14.42 % of total
Top 10 words :
[('106', 10.298795943709614), ('10s', 10.298795943709614), ('111',
10.298795943709614), ('126', 10.298795943709614), ('128', 10.298795943709614),
('148', 10.298795943709614), ('15am', 10.298795943709614), ('185',
10.298795943709614), ('190', 10.298795943709614), ('1907', 10.298795943709614)]
********************************************************************************
********************
Bin :  20
Range : 10.45 - 11.0
No of words : 6426.0 which is 26.32 % of total
Top 10 words :
```

```
[('04', 10.858411731645036), ('08', 10.858411731645036), ('0ver',
10.858411731645036), ('109', 10.858411731645036), ('10x10', 10.858411731645036),
('116', 10.858411731645036), ('121', 10.858411731645036), ('129',
10.858411731645036), ('12pm', 10.858411731645036), ('144', 10.858411731645036)]
********************************************************************************
********************
```

[0]:
```
# we clearly see that the first 6 bins have very few words and the last bin has
# words which are mostly digits in string format which do not add much meaning

# llimit, hlimit = 3.3, 9.6
llimit, hlimit = 2, 10

def exclude_words(text):
  return ' '.join([word for word in text.split() if \
                    feature_Idf_Map.get(word) is None or
                    (feature_Idf_Map.get(word) and \
                  (feature_Idf_Map.get(word)>=llimit and \
                      feature_Idf_Map.get(word)<=hlimit))])
```

[0]:
```
# changing the essay text to exclude the extra words

dfX_train['mod_essay']=dfX_train.essay.apply(exclude_words)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[0]:
```
dfX_test['mod_essay']=dfX_test.essay.apply(exclude_words)
```

[0]:
```
dfX_cv['mod_essay']=dfX_cv.essay.apply(exclude_words)
```

[0]:
```
tokenizer_Essay2 = Tokenizer(oov_token='<oov>')
tokenizer_Essay2.fit_on_texts(dfX_train.mod_essay.to_list())
```

[0]:
```
tokenised_essay2_train = tokenizer_Essay2.texts_to_sequences(dfX_train.
 ↪mod_essay)
tokenised_essay2_cv = tokenizer_Essay2.texts_to_sequences(dfX_cv.mod_essay)
tokenised_essay2_test = tokenizer_Essay2.texts_to_sequences(dfX_test.mod_essay)
X_train_essay2 = pad_sequences(tokenised_essay2_train, maxlen=max_essay_length)
X_test_essay2 = pad_sequences(tokenised_essay2_test, maxlen=max_essay_length)
X_cv_essay2 = pad_sequences(tokenised_essay2_cv, maxlen=max_essay_length)
```

[0]:
```
EMBEDDING_DIMS = 300     # glove vectors are 300 dims
VOCAB_SIZE = len(list(tokenizer_Essay2.word_counts.keys()))
embedding_matrix2 = np.zeros((VOCAB_SIZE+1, EMBEDDING_DIMS))
for word, i in tokenizer_Essay2.word_index.items():
    embedding_vector = glove.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix2[i-1] = embedding_vector
```

[0]:
```
# Modified essay
mod_essay_Inp = Input(shape=(max_essay_length,),dtype='int32',
                      name='mod_essay_Inp')
embedded_mod_Essay = Embedding(input_dim=len(tokenizer_Essay2.word_index.
 ↪items()),
                        output_dim=300,name='embedded_mod_Essay',
                        weights=[embedding_matrix2],
                        trainable=False)(mod_essay_Inp)
mod_essay_LSTM = LSTM(units=128, return_sequences=True)(embedded_mod_Essay)
mod_essay_Out = Flatten()(mod_essay_LSTM)
```

[0]:
```
# concatenating all the outputs

concatenated_Outs2 = concatenate([school_state_out, teacher_Pref_out,
                pgCategory_Out,cleanCategory_Out,clean_subcategories_Out,
                mod_essay_Out,remaining_cols_Out])
```

[0]:
```
outPut = Dense(128,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(concatenated_Outs2)
outPut = Dropout(0.4)(outPut)
outPut = Dense(64,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(outPut)
outPut = Dropout(0.4)(outPut)
outPut = BatchNormalization()(outPut)
outPut = Dense(32,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(outPut)
outPut = Dropout(0.4)(outPut)
outPut = Dense(2, activation = 'softmax')(outPut)
```

[0]:
```
# create model with all the previously defined inputs
model2 = Model([school_state_inp,teacher_Pref_inp,pgCategory_Inp,
                cleanCategory_Inp,clean_subcategories_Inp,mod_essay_Inp,
                remaining_cols_Inp], outPut)
model2.compile(loss='categorical_crossentropy',
               optimizer=optimizers.Adam(lr=0.0006,decay = 1e-4),
               metrics=[auc])
print(model2.summary())
```

```
Model: "model_4"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
mod_essay_Inp (InputLayer)      [(None, 350)]        0
________________________________________________________________________________
__________________
school_state_inp (InputLayer)   [(None, 1)]          0
________________________________________________________________________________
__________________
teacher_Pref_inp (InputLayer)   [(None, 1)]          0
________________________________________________________________________________
__________________
pgCategory_Inp (InputLayer)     [(None, 1)]          0
________________________________________________________________________________
__________________
cleanCategory_Inp (InputLayer)  [(None, 3)]          0
________________________________________________________________________________
__________________
clean_subcategories_Inp (InputL [(None, 3)]          0
________________________________________________________________________________
__________________
embedded_mod_Essay (Embedding)  (None, 350, 300)     12005100
mod_essay_Inp[0][0]
________________________________________________________________________________
__________________
embedded_school_state (Embeddin (None, 1, 6)         312
school_state_inp[0][0]
________________________________________________________________________________
__________________
embedded_teacher_Pref (Embeddin (None, 1, 2)         12
teacher_Pref_inp[0][0]
________________________________________________________________________________
__________________
embedded_pgCategory (Embedding) (None, 1, 2)         10
pgCategory_Inp[0][0]
________________________________________________________________________________
__________________
embedded_cleanCategory (Embeddi (None, 3, 3)         30
cleanCategory_Inp[0][0]
________________________________________________________________________________
__________________
embedded_cleanSubCategory (Embe (None, 3, 5)         155
clean_subcategories_Inp[0][0]
________________________________________________________________________________
__________________
```

```
lstm_1 (LSTM)                  (None, 350, 128)     219648
embedded_mod_Essay[0][0]

--------------------------------------------------------------------------------
------------------
remaining_cols_Inp (InputLayer) [(None, 2)]         0

--------------------------------------------------------------------------------
------------------
flatten (Flatten)              (None, 6)            0
embedded_school_state[0][0]

--------------------------------------------------------------------------------
------------------
flatten_1 (Flatten)            (None, 2)            0
embedded_teacher_Pref[0][0]

--------------------------------------------------------------------------------
------------------
flatten_2 (Flatten)            (None, 2)            0
embedded_pgCategory[0][0]

--------------------------------------------------------------------------------
------------------
flatten_3 (Flatten)            (None, 9)            0
embedded_cleanCategory[0][0]

--------------------------------------------------------------------------------
------------------
flatten_4 (Flatten)            (None, 15)           0
embedded_cleanSubCategory[0][0]

--------------------------------------------------------------------------------
------------------
flatten_6 (Flatten)            (None, 44800)        0           lstm_1[0][0]

--------------------------------------------------------------------------------
------------------
dense (Dense)                  (None, 16)           48
remaining_cols_Inp[0][0]

--------------------------------------------------------------------------------
------------------
concatenate_1 (Concatenate)    (None, 44850)        0           flatten[0][0]
                                                                flatten_1[0][0]
                                                                flatten_2[0][0]
                                                                flatten_3[0][0]
                                                                flatten_4[0][0]
                                                                flatten_6[0][0]
                                                                dense[0][0]

--------------------------------------------------------------------------------
------------------
dense_5 (Dense)                (None, 128)          5740928
concatenate_1[0][0]

--------------------------------------------------------------------------------
------------------
dropout_3 (Dropout)            (None, 128)          0           dense_5[0][0]
```

```
________________________________________________________________________________
__________________
dense_6 (Dense)                 (None, 64)           8256        dropout_3[0][0]
________________________________________________________________________________
__________________
dropout_4 (Dropout)             (None, 64)           0           dense_6[0][0]
________________________________________________________________________________
__________________
batch_normalization_1 (BatchNor (None, 64)           256         dropout_4[0][0]
________________________________________________________________________________
__________________
dense_7 (Dense)                 (None, 32)           2080
batch_normalization_1[0][0]
________________________________________________________________________________
__________________
dropout_5 (Dropout)             (None, 32)           0           dense_7[0][0]
________________________________________________________________________________
__________________
dense_8 (Dense)                 (None, 2)            66          dropout_5[0][0]
================================================================================
==================
Total params: 17,976,901
Trainable params: 5,971,673
Non-trainable params: 12,005,228
________________________________________________________________________________
__________________
None
```

[0]:
```
# Lets run the model
filepath=dir_path+"best_weights2.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc',
                    verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]

if os.path.isfile(dir_path+'best_weights2.hdf5'):
  model2.load_weights(dir_path+'best_weights2.hdf5')

model2.fit([X_train_schoolState,X_train_teacherPrefix,X_train_pgCategory,
           X_train_cleanCategory,X_train_cleanSubCategory,X_train_essay2,
           np.array(dfX_train['remaining_cols'].to_list())],␣
 ↪to_categorical(y_train),
          ␣
 ↪epochs=20,verbose=2,batch_size=256,validation_data=([X_cv_schoolState,
           X_cv_teacherPrefix,X_cv_pgCategory,X_cv_cleanCategory,
           X_cv_cleanSubCategory,X_cv_essay2,np.array(dfX_cv['remaining_cols'].
 ↪to_list())]
           ,to_categorical(y_cv)),
            callbacks =callbacks_list)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/math_grad.py:1424: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 76473 samples, validate on 16387 samples
Epoch 1/20

Epoch 00001: val_auc improved from -inf to 0.59871, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 146s - loss: 0.7770 - auc: 0.5182 - val_loss: 0.5696 - val_auc:
0.5987
Epoch 2/20

Epoch 00002: val_auc improved from 0.59871 to 0.60677, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 143s - loss: 0.5873 - auc: 0.5376 - val_loss: 0.5348 - val_auc:
0.6068
Epoch 3/20

Epoch 00003: val_auc improved from 0.60677 to 0.60850, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 143s - loss: 0.5357 - auc: 0.5562 - val_loss: 0.5063 - val_auc:
0.6085
Epoch 4/20

Epoch 00004: val_auc did not improve from 0.60850
76473/76473 - 141s - loss: 0.5108 - auc: 0.5605 - val_loss: 0.4931 - val_auc:
0.6084
Epoch 5/20

Epoch 00005: val_auc improved from 0.60850 to 0.61756, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 141s - loss: 0.4891 - auc: 0.5759 - val_loss: 0.4779 - val_auc:
0.6176
Epoch 6/20

Epoch 00006: val_auc did not improve from 0.61756
76473/76473 - 139s - loss: 0.4749 - auc: 0.5861 - val_loss: 0.4725 - val_auc:
0.6073
```

```
Epoch 7/20

Epoch 00007: val_auc improved from 0.61756 to 0.65281, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 140s - loss: 0.4650 - auc: 0.5957 - val_loss: 0.4491 - val_auc:
0.6528
Epoch 8/20

Epoch 00008: val_auc did not improve from 0.65281
76473/76473 - 139s - loss: 0.4577 - auc: 0.5887 - val_loss: 0.4431 - val_auc:
0.6522
Epoch 9/20

Epoch 00009: val_auc improved from 0.65281 to 0.68000, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 140s - loss: 0.4438 - auc: 0.6382 - val_loss: 0.4328 - val_auc:
0.6800
Epoch 10/20

Epoch 00010: val_auc improved from 0.68000 to 0.69455, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 140s - loss: 0.4370 - auc: 0.6380 - val_loss: 0.4252 - val_auc:
0.6945
Epoch 11/20

Epoch 00011: val_auc improved from 0.69455 to 0.69838, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 140s - loss: 0.4231 - auc: 0.6822 - val_loss: 0.4187 - val_auc:
0.6984
Epoch 12/20

Epoch 00012: val_auc improved from 0.69838 to 0.69954, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 139s - loss: 0.4207 - auc: 0.6826 - val_loss: 0.4164 - val_auc:
0.6995
Epoch 13/20

Epoch 00013: val_auc improved from 0.69954 to 0.71383, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 138s - loss: 0.4117 - auc: 0.6993 - val_loss: 0.4082 - val_auc:
0.7138
Epoch 14/20
Epoch 00014: val_auc improved from 0.71383 to 0.72182, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 139s - loss: 0.4035 - auc: 0.7158 - val_loss: 0.4032 - val_auc:
0.7218
Epoch 15/20

Epoch 00015: val_auc did not improve from 0.72182
76473/76473 - 137s - loss: 0.3971 - auc: 0.7273 - val_loss: 0.4019 - val_auc:
0.7211
Epoch 16/20

Epoch 00016: val_auc improved from 0.72182 to 0.72828, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights2.hdf5
76473/76473 - 138s - loss: 0.3892 - auc: 0.7421 - val_loss: 0.4067 - val_auc:
0.7283
Epoch 17/20

Epoch 00017: val_auc did not improve from 0.72828
76473/76473 - 137s - loss: 0.3824 - auc: 0.7537 - val_loss: 0.3956 - val_auc:
0.7207
Epoch 18/20

Epoch 00018: val_auc did not improve from 0.72828
76473/76473 - 137s - loss: 0.3761 - auc: 0.7647 - val_loss: 0.4018 - val_auc:
0.7276
Epoch 19/20

Epoch 00019: val_auc did not improve from 0.72828
76473/76473 - 136s - loss: 0.3717 - auc: 0.7730 - val_loss: 0.3956 - val_auc:
0.7233
Epoch 20/20

Epoch 00020: val_auc did not improve from 0.72828
76473/76473 - 136s - loss: 0.3642 - auc: 0.7845 - val_loss: 0.3968 - val_auc:
0.7233
```

[0]:
```
<tensorflow.python.keras.callbacks.History at 0x7f134b3b2fd0>
```

[0]:
```
# AUC for test data
print("Test AUC for Model2 : %0.3f"%roc_auc_score(to_categorical(y_test),
      model2.predict([X_test_schoolState,X_test_teacherPrefix,X_test_pgCategory,
      X_test_cleanCategory,X_test_cleanSubCategory,
      X_test_essay2,np.array(dfX_test['remaining_cols'].to_list())]),
      average='weighted'))
      # Last 0.683
```

```
Test AUC for Model2 : 0.718
```

[0]:
```
# AUC for test data
print("Test AUC for Model2 : %0.3f"%roc_auc_score(to_categorical(y_test),
      model2.predict([X_test_schoolState,X_test_teacherPrefix,X_test_pgCategory,
      X_test_cleanCategory,X_test_cleanSubCategory,
      X_test_essay2,np.array(dfX_test['remaining_cols'].to_list())]),
      average='micro'))

# 0.876
```

```
Test AUC for Model2 : 0.905
```

[0]:
```
table.add_row(['Model 2',0.718,0.905])
```

0.1.5 Model-3

ref: https://i.imgur.com/fkQ8nGo.png

- **input_seq_total_text_data**:
- **Other_than_text_data**:
  . Convert all your Categorical values to onehot coded and then concatenate all these onehot vectors . Neumerical values and use CNN1D as shown in above figure. . You are free to choose all CNN parameters like kernel sizes, stride.

[45]:
```
# We will use the 'Essay' layer from the Model1
# Lets work on the all other columns

mlbinarizer_schoolState = MultiLabelBinarizer()
mlbinarizer_schoolState.fit([dfX_train.school_state.to_list()])
```

[45]:
```
MultiLabelBinarizer(classes=None, sparse_output=False)
```

[0]:
```
X_train_schoolState = mlbinarizer_schoolState.transform(\
                        dfX_train.school_state.apply(lambda x : [x]))
X_test_schoolState = mlbinarizer_schoolState.transform(\
                        dfX_test.school_state.apply(lambda x : [x]))
X_cv_schoolState = mlbinarizer_schoolState.transform(\
                        dfX_cv.school_state.apply(lambda x : [x]))
```

[47]:
```
print(X_train_schoolState.shape)
```

```
(76473, 51)
```

[0]:
```
mlbinarizer_teacherPrefix = MultiLabelBinarizer()
mlbinarizer_teacherPrefix.fit([dfX_train.teacher_prefix.to_list()])

X_train_teacherPrefix = mlbinarizer_teacherPrefix.transform(\
                      dfX_train.teacher_prefix.apply(lambda x : [x]))
X_test_teacherPrefix = mlbinarizer_teacherPrefix.transform(\
                      dfX_test.teacher_prefix.apply(lambda x : [x]))
X_cv_teacherPrefix = mlbinarizer_teacherPrefix.transform(\
                      dfX_cv.teacher_prefix.apply(lambda x : [x]))
```

[49]:
```
print(X_train_teacherPrefix.shape)
```

```
(76473, 5)
```

[0]:
```
mlbinarizer_pgCategory = MultiLabelBinarizer()
mlbinarizer_pgCategory.fit([dfX_train.project_grade_category.to_list()])

X_train_pgCategory = mlbinarizer_pgCategory.transform(\
                      dfX_train.project_grade_category.apply(lambda x : [x]))
X_test_pgCategory = mlbinarizer_pgCategory.transform(\
                      dfX_test.project_grade_category.apply(lambda x : [x]))
X_cv_pgCategory = mlbinarizer_pgCategory.transform(\
                      dfX_cv.project_grade_category.apply(lambda x : [x]))
```

[51]:
```
print(X_train_pgCategory.shape)
```

```
(76473, 4)
```

[0]:
```
mlbinarizer_cleanCategory = MultiLabelBinarizer()
mlbinarizer_cleanCategory.fit([dfX_train.clean_categories.to_list()])

X_train_cleanCategory = mlbinarizer_cleanCategory.transform(\
                      dfX_train.clean_categories.apply(lambda x : [x]))
X_test_cleanCategory = mlbinarizer_cleanCategory.transform(\
                      dfX_test.clean_categories.apply(lambda x : [x]))
X_cv_cleanCategory = mlbinarizer_cleanCategory.transform(\
                      dfX_cv.clean_categories.apply(lambda x : [x]))
```

[53]:
```
print(X_train_cleanCategory.shape)
```

```
(76473, 51)
```

[54]:
```
mlbinarizer_cleanSubCategory = MultiLabelBinarizer()
mlbinarizer_cleanSubCategory.fit([dfX_train.clean_subcategories.to_list()])

X_train_cleanSubCategory = mlbinarizer_cleanSubCategory.transform(\
                      dfX_train.clean_subcategories.apply(lambda x : [x]))
X_test_cleanSubCategory = mlbinarizer_cleanSubCategory.transform(\
                      dfX_test.clean_subcategories.apply(lambda x : [x]))
X_cv_cleanSubCategory = mlbinarizer_cleanSubCategory.transform(\
                      dfX_cv.clean_subcategories.apply(lambda x : [x]))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/label.py:951:
UserWarning: unknown class(es) ['civics_government parentinvolvement',
'civics_government teamsports', 'earlydevelopment history_geography', 'economics
foreignlanguages', 'financialliteracy health_lifescience', 'socialsciences
teamsports'] will be ignored
  .format(sorted(unknown, key=str)))
/usr/local/lib/python3.6/dist-packages/sklearn/preprocessing/label.py:951:
UserWarning: unknown class(es) ['civics_government teamsports', 'economics
music', 'economics other', 'financialliteracy health_lifescience',
'financialliteracy health_wellness', 'socialsciences teamsports'] will be
ignored
  .format(sorted(unknown, key=str)))
```

[55]:
```
print(X_train_cleanSubCategory.shape)
```

```
(76473, 392)
```

[0]:
```
X_train_concat = np.
 ↪hstack((X_train_schoolState,X_train_teacherPrefix,X_train_pgCategory,
          X_train_cleanCategory,X_train_cleanSubCategory))
X_test_concat = np.
 ↪hstack((X_test_schoolState,X_test_teacherPrefix,X_test_pgCategory,
          X_test_cleanCategory,X_test_cleanSubCategory))
X_cv_concat = np.hstack((X_cv_schoolState,X_cv_teacherPrefix,X_cv_pgCategory,
          X_cv_cleanCategory,X_cv_cleanSubCategory))
```

[0]:
```
X_train_concat=np.expand_dims(X_train_concat, axis=2)
X_test_concat=np.expand_dims(X_test_concat, axis=2)
X_cv_concat=np.expand_dims(X_cv_concat, axis=2)
```

[58]:
```
X_train_concat.shape
```

[58]:
```
(76473, 503, 1)
```

[0]:
```
dims = X_train_concat.shape[1]
```

[0]:
```
stacked_Inp = Input(shape=(dims,1), name='stacked_Inp')
conv_out = Conv1D(128, kernel_size=(3),activation='relu')(stacked_Inp)  #128
conv_out = Conv1D(64, kernel_size=(2),activation='relu')(conv_out)  # 64
# conv_out = Conv1D(32, kernel_size=(2),activation='relu')(conv_out)   ␣
 ↪Ommiting
#                  this layer
conv_out = Flatten()(conv_out)
```

[0]:
```
concatenated_Outs3=concatenate([essay_Out, conv_out])
```

[0]:
```
# We are using the same configuration as the model1

outPut = Dense(256,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(concatenated_Outs3)
outPut = Dropout(0.5)(outPut)
outPut = Dense(64,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(outPut)
outPut = Dropout(0.5)(outPut)
outPut = BatchNormalization()(outPut)
outPut = Dense(32,activation='relu',
               kernel_initializer=initializers.he_normal(),
               kernel_regularizer=regularizers.l2(0.001))(outPut)
outPut = Dropout(0.5)(outPut)
outPut = Dense(2, activation = 'softmax')(outPut)
```

[71]:
```
# create model with all the previously defined inputs
model3 = Model([essay_Inp, stacked_Inp], outPut)
model3.compile(loss='categorical_crossentropy',
               optimizer=optimizers.Adam(lr=0.0006,decay = 1e-4),
               metrics=[auc])
print(model3.summary())
```

```
Model: "model_1"
________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
================================================================================
==================
essay_Inp (InputLayer)          [(None, 350)]        0

________________________________________________________________________________
__________________
stacked_Inp (InputLayer)        [(None, 503, 1)]     0

________________________________________________________________________________
__________________
embedded_Essay (Embedding)      (None, 350, 300)     14799000    essay_Inp[0][0]

________________________________________________________________________________
__________________
conv1d_2 (Conv1D)               (None, 501, 128)     512
stacked_Inp[0][0]
________________________________________________________________________________
__________________
lstm (LSTM)                     (None, 350, 128)     219648
embedded_Essay[0][0]
________________________________________________________________________________
__________________
conv1d_3 (Conv1D)               (None, 500, 64)      16448       conv1d_2[0][0]
```

```
--------------------------------------------------------------------------------
------------------
flatten_5 (Flatten)             (None, 44800)        0           lstm[0][0]

--------------------------------------------------------------------------------
------------------
flatten_7 (Flatten)             (None, 32000)        0           conv1d_3[0][0]

--------------------------------------------------------------------------------
------------------
concatenate_2 (Concatenate)     (None, 76800)        0           flatten_5[0][0]
                                                                 flatten_7[0][0]

--------------------------------------------------------------------------------
------------------
dense_9 (Dense)                 (None, 256)          19661056
concatenate_2[0][0]

--------------------------------------------------------------------------------
------------------
dropout_6 (Dropout)             (None, 256)          0           dense_9[0][0]

--------------------------------------------------------------------------------
------------------
dense_10 (Dense)                (None, 64)           16448       dropout_6[0][0]

--------------------------------------------------------------------------------
------------------
dropout_7 (Dropout)             (None, 64)           0           dense_10[0][0]

--------------------------------------------------------------------------------
------------------
batch_normalization_2 (BatchNor (None, 64)           256         dropout_7[0][0]

--------------------------------------------------------------------------------
------------------
dense_11 (Dense)                (None, 32)           2080
batch_normalization_2[0][0]

--------------------------------------------------------------------------------
------------------
dropout_8 (Dropout)             (None, 32)           0           dense_11[0][0]

--------------------------------------------------------------------------------
------------------
dense_12 (Dense)                (None, 2)            66          dropout_8[0][0]
================================================================================
==================
Total params: 34,715,514
Trainable params: 19,916,386
Non-trainable params: 14,799,128

--------------------------------------------------------------------------------
------------------
None
```

[72]:
```
X_cv_essay.shape
```

[72]:
```
(16387, 350)
```

[74]:
```
# Lets run the model
filepath=dir_path+"best_weights3.hdf5"
checkpoint = ModelCheckpoint(filepath, monitor='val_auc',
                     verbose=1, save_best_only=True, mode='max')
callbacks_list = [checkpoint]

if os.path.isfile(dir_path+'best_weights3.hdf5'):
  model3.load_weights(dir_path+'best_weights3.hdf5')

model3.fit([X_train_essay, X_train_concat], to_categorical(y_train),
           epochs=5,verbose=2,batch_size=256,
           validation_data=([X_cv_essay,X_cv_concat],to_categorical(y_cv)),
           callbacks =callbacks_list)
```

```
Train on 76473 samples, validate on 16387 samples
Epoch 1/5

Epoch 00001: val_auc improved from -inf to 0.71909, saving model to
/content/drive/My Drive/Colab
Notebooks/AppliedAI/LSTM_Donors_Choose/best_weights3.hdf5
76473/76473 - 146s - loss: 0.4252 - auc: 0.7538 - val_loss: 0.4245 - val_auc:
0.7191
Epoch 2/5

Epoch 00002: val_auc did not improve from 0.71909
76473/76473 - 141s - loss: 0.4152 - auc: 0.7684 - val_loss: 0.4411 - val_auc:
0.7156
Epoch 3/5

Epoch 00003: val_auc did not improve from 0.71909
76473/76473 - 140s - loss: 0.4072 - auc: 0.7891 - val_loss: 0.4382 - val_auc:
0.7171
Epoch 4/5

Epoch 00004: val_auc did not improve from 0.71909
76473/76473 - 139s - loss: 0.3995 - auc: 0.8046 - val_loss: 0.4441 - val_auc:
0.7080
Epoch 5/5

Epoch 00005: val_auc did not improve from 0.71909
76473/76473 - 139s - loss: 0.3859 - auc: 0.8274 - val_loss: 0.4676 - val_auc:
0.7025
```

[74]:
```
<tensorflow.python.keras.callbacks.History at 0x7ffb4efee748>
```

[75]:
```
# AUC for test data
print("Test AUC for Model3 : %0.3f"%roc_auc_score(to_categorical(y_test),
      model3.predict([X_test_essay, X_test_concat]),
        average='weighted'))
```

```
Test AUC for Model3 : 0.707
```

[76]:
```
# AUC for test data
print("Test AUC for Model3 : %0.3f"%roc_auc_score(to_categorical(y_test),
        model3.predict([X_test_essay, X_test_concat]),
        average='micro'))
```

```
Test AUC for Model3 : 0.902
```

[0]:
```
table.add_row(['Model 3',0.707,0.902])
```

[8]:
```
print(table)
```

```
+---------+-------------------+----------------+
|  Model  | Weighted Test AUC | Micro Test AUC |
+---------+-------------------+----------------+
| Model 1 |       0.762       |     0.916      |
| Model 2 |       0.718       |     0.905      |
| Model 3 |       0.707       |     0.902      |
+---------+-------------------+----------------+
```

[0]:
```

```