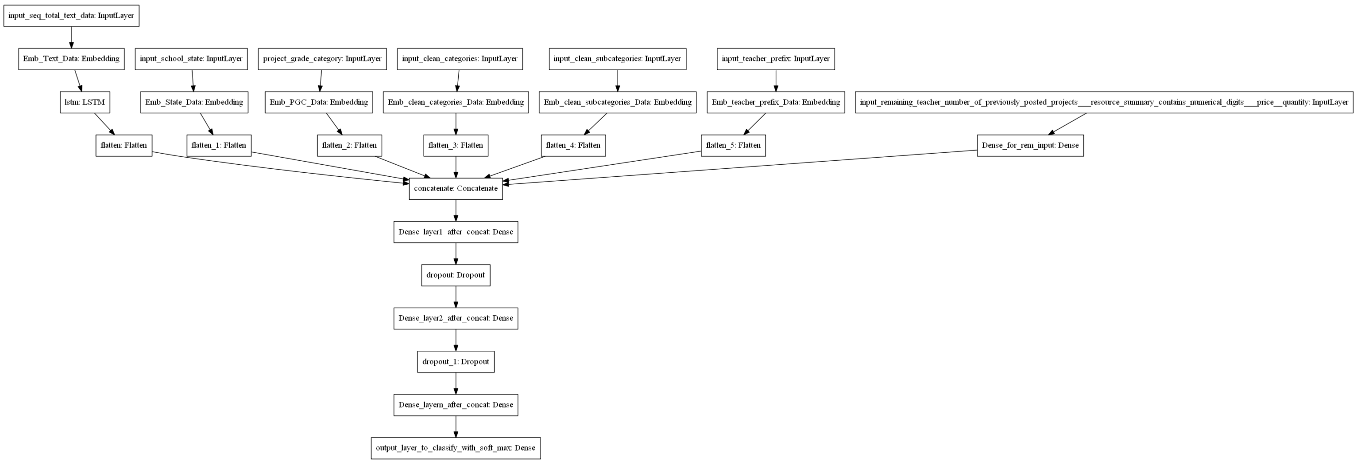
# Assignment : 14

1. Preprocess all the Data we have in DonorsChoose Dataset (https://drive.google.com/drive/folders/1MIwK7BQMev8f5CbDDVNLPaFGB32pFN60) use train.csv
2. Combine 4 essay's into one column named - 'preprocessed_essays'.
3. After step 2 you have to train 3 types of models as discussed below.
4. For all the model use 'auc' (https://scikit-learn.org/stable/modules/model_evaluation.html#roc-metrics) as a metric. check this (https://datascience.stackexchange.com/a/20192) for using auc as a metric
5. You are free to choose any number of layers/hiddden units but you have to use same type of architectures shown below.
6. You can use any one of the optimizers and choice of Learning rate and momentum, resources: cs231n class notes (http://cs231n.github.io/neural-networks-3/), cs231n class video (https://www.youtube.com/watch?v=hd_KFJ5ktUc).
7. For all the model's use TensorBoard (https://www.youtube.com/watch?v=2U6Jl7oqRkM) and plot the Metric value and Loss with epoch. While submitting, take a screenshot of plots and include those images in .ipynb notebook and PDF.
8. Use Categorical Cross Entropy as Loss to minimize.

## Model-1

Build and Train deep neural network as shown below


input_seq_total_text_data: InputLayer
Emb_Text_Data: Embedding
input_school_state: InputLayer
project_grade_category: InputLayer
input_clean_categories: InputLayer
input_clean_subcategories: InputLayer
input_teacher_prefix: InputLayer
lstm: LSTM
Emb_State_Data: Embedding
Emb_PGC_Data: Embedding
Emb_clean_categories_Data: Embedding
Emb_clean_subcategories_Data: Embedding
Emb_teacher_prefix_Data: Embedding
input_remaining_teacher_number_of_previously_posted_projects___resource_summary_contains_numerical_digits___price__quantity: InputLayer
flatten: Flatten
flatten_1: Flatten
flatten_2: Flatten
flatten_3: Flatten
flatten_4: Flatten
flatten_5: Flatten
Dense_for_rem_input: Dense
concatenate: Concatenate
Dense_layer1_after_concat: Dense
dropout: Dropout
Dense_layer2_after_concat: Dense
dropout_1: Dropout
Dense_layern_after_concat: Dense
output_layer_to_classify_with_soft_max: Dense

ref: https://i.imgur.com/w395Yk9.png (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_** ---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.

In [2]:

```
# 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)
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-2-ed7dba31d057> in <module>()
      1 # https://stats.stackexchange.com/questions/270546/how-does-keras-
embedding-layer-work
----> 2 input_layer = Input(shape=(n,))
      3 embedding = Embedding(no_1, no_2, input_length=n)(input_layer)
      4 flatten = Flatten()(embedding)

NameError: name 'Input' is not defined
```

In [3]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

### 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/ (https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/)

### 2. Please go through this link https://keras.io/getting-started/functional-api-guide/ (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.

In [4]:

```
import numpy as np
import pandas as pd
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Input , Dropout
from keras.layers import Flatten
from keras.layers import concatenate
from keras.layers.embeddings import Embedding
from keras.models import Model
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.preprocessing.text import Tokenizer
import matplotlib.pyplot as plt
import pickle
from keras.layers import LSTM
from keras.preprocessing.text import text_to_word_sequence
import tensorflow as tf
from keras.callbacks import ModelCheckpoint,TensorBoard,ReduceLROnPlateau, EarlyStopping
from keras.layers.normalization import BatchNormalization
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
from sklearn.metrics import roc_auc_score
from keras.models import load_model
import tensorflow as tf
```

In [5]:

```
import pandas as pd

data = pd.read_csv("/content/drive/My Drive/9_Donors_choose_DT/preprocessed_data.csv")
data.head()
```

Out[5]:

| | school_state | teacher_prefix | project_grade_category | teacher_number_of_previously_posted_ |
|---|---|---|---|---|
| **0** | ca | mrs | grades_prek_2 | |
| **1** | ut | ms | grades_3_5 | |
| **2** | ca | mrs | grades_prek_2 | |
| **3** | ga | mrs | grades_prek_2 | |
| **4** | wa | mrs | grades_3_5 | |

In [6]:

```
data.columns
```

Out[6]:

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

In [7]:

```
y=data["project_is_approved"].values
```

In [8]:

```
data.drop("project_is_approved",axis = 1, inplace = True)
```

## FOr clean Subcategories

In [9]:

```
d=[]
data['temp']=data['clean_subcategories'].apply(lambda x: x.split()).apply(lambda x: d+x
)
```

In [10]:

```
data['temp']
```

Out[10]:

```
0            [appliedsciences, health_lifescience]
1                                  [specialneeds]
2                                      [literacy]
3                              [earlydevelopment]
4                                      [literacy]
                            ...
109243                               [teamsports]
109244                  [earlydevelopment, other]
109245    [appliedsciences, environmentalscience]
109246                       [health_lifescience]
109247              [literacy, literature_writing]
Name: temp, Length: 109248, dtype: object
```

In [11]:

```
d=[]
data['clean_subcategories'].apply(lambda x: x.split()).apply(lambda x: d.extend(x))
z=list(set(d))
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(z)
```

Out[11]:

```
LabelEncoder()
```

In [12]:

```
data['clean_subcategories_label']=data['temp'].apply(lambda q:label.transform(q))
```

In [13]:

```
data['clean_subcategories_label']
```

Out[13]:

```
0          [0, 14]
1             [26]
2             [17]
3              [6]
4             [17]
            ...   
109243        [27]
109244     [6, 22]
109245      [0, 8]
109246        [14]
109247    [17, 18]
Name: clean_subcategories_label, Length: 109248, dtype: object
```

In [14]:

```
np.max(data['clean_subcategories_label'].apply(lambda x: len(x)))
```

Out[14]:

```
3
```

In [15]:

```
from keras.preprocessing import sequence
max_review_length = 3
clean_subcategories_label_= sequence.pad_sequences(data['clean_subcategories_label'].va
lues, maxlen=max_review_length, padding='post')
```

In [16]:

```
import scipy.sparse as sparse
arr = sparse.coo_matrix(clean_subcategories_label_, shape=(109248,3))
data['clean_subcategories_label_'] = arr.toarray().tolist()
print(data)
```

```
       school_state  ... clean_subcategories_label_
0                ca  ...                [0, 14, 0]
1                ut  ...                [26, 0, 0]
2                ca  ...                [17, 0, 0]
3                ga  ...                 [6, 0, 0]
4                wa  ...                [17, 0, 0]
...             ...  ...                       ...
109243           hi  ...                [27, 0, 0]
109244           nm  ...                [6, 22, 0]
109245           il  ...                 [0, 8, 0]
109246           hi  ...                [14, 0, 0]
109247           ca  ...               [17, 18, 0]

[109248 rows x 11 columns]
```

In [17]:

```
data.shape
```

Out[17]:

```
(109248, 11)
```

In [18]:

```
data.drop(['temp','clean_subcategories_label'],axis=1,inplace=True)
```

In [19]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(data, y, test_size=0.33, stratify=y
)



print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
(73196, 9) (73196,)
(36052, 9) (36052,)
```

# Tokenizer

In [20]:

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words= 1000000 )
tokenizer.fit_on_texts(X_train["essay"])
```

In [21]:

```
X_train['es_tok']  = tokenizer.texts_to_sequences(X_train['essay'].values)

X_test['es_tok']   = tokenizer.texts_to_sequences(X_test['essay'].values)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:1: SettingWit
hCopyWarning:
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: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: SettingWit
hCopyWarning:
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: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing import
s until
```

In [22]:

```
X_test.head()
```

Out[22]:

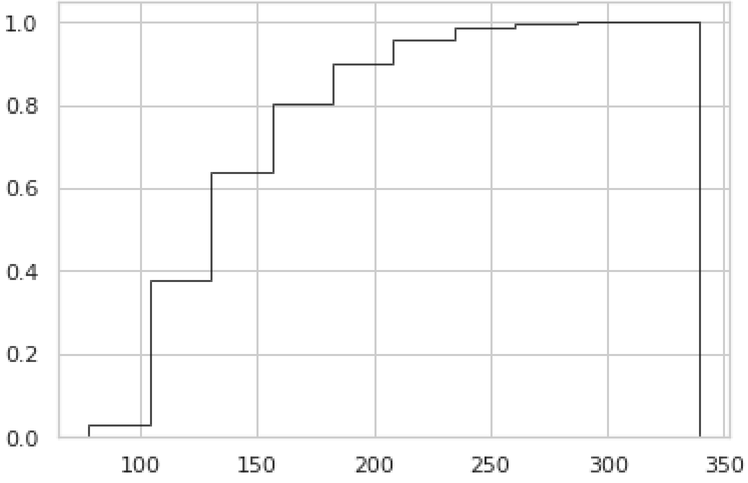
| | school_state | teacher_prefix | project_grade_category | teacher_number_of_previously_p |
|---|---|---|---|---|
| **60249** | ga | ms | grades_prek_2 | |
| **105342** | tn | mrs | grades_prek_2 | |
| **43904** | ct | mrs | grades_prek_2 | |
| **108262** | hi | mrs | grades_3_5 | |
| **19282** | wa | mrs | grades_prek_2 | |

In [23]:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(style="whitegrid")
plt.hist(X_train['es_tok'].apply(lambda x: len(x)),cumulative=True, density=True,label=
'CDF', alpha=0.8, color='k',histtype='step')
#np.max(X_train['es_tok'].apply(lambda x: len(x)))
```

Out[23]:

```
(array([0.03113558, 0.37738401, 0.63805399, 0.80541286, 0.90078693,
        0.9558036 , 0.9848489 , 0.99678944, 0.99969944, 1.        ]),
 array([ 78. , 104.1, 130.2, 156.3, 182.4, 208.5, 234.6, 260.7, 286.8,
        312.9, 339. ]),
 <a list of 1 Patch objects>)
```



## Percentile length of review which covers almost 98% is 250, we use Padding of 250

In [24]:

```
from keras.preprocessing import sequence
max_review_length = 250
X_train_pad = sequence.pad_sequences(X_train['es_tok'].values, maxlen=max_review_length
)

X_test_pad  = sequence.pad_sequences(X_test['es_tok'].values, maxlen=max_review_length)
```

In [25]:

```
X_train_pad[0]
```

Out[25]:

```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,    25,     3,
         350,   268,    23,   417,     1,   140,    38,    63,   103,
        1403,    12,    52,   140,   188,    58,   745,    14,   230,
          14,   891,  6035,     1,    33,  1040,    39,  2201,   327,
         177,  2720,   246,   750,     1,   170,    33,  1109,    87,
         462,    53,   230,    14,   171,     8,   122,    40,   412,
         122,   317,    25,   145,  1724,   299,   522,   652,   894,
        1081,   193,  1015,   170,   327,    38,   222,     8,   255,
        1121,    26,    14,    18,    96,   392,    46,   912,  2768,
         312,    71,   774,     1,  1005,    34,    10,    38,   222,
         475,    31,  1276,     5,   395,   527,     1, 12868,   306,
           1,   266,   200,   150,    18,   208,    15,    11,   815,
           1,    15,   882,    67,   684,   403,    71,    74,  3333,
         152,   389,   101,   326,  2408,    55,    13], dtype=int32)
```

In [26]:

```
import pickle
with open('/content/drive/My Drive/9_Donors_choose_DT/glove_vectors', 'rb') as f:
    glove = pickle.load(f)
    glove_words =  set(glove.keys())
```

In [27]:

```
len(tokenizer.word_index.items())
```

Out[27]:

```
48129
```

In [28]:

```
l=tokenizer.word_index.items()
number_of_words_in_corpus = len(tokenizer.word_index)

embedding_matrix = np.zeros((number_of_words_in_corpus+1, 300))
for word, i in l:


  if word in glove_words:
    embedding_matrix[i] =glove[word]
```

In [29]:

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(data['school_state'].values)


X_train_school_state_label_encoding = label.transform(X_train['school_state'].values)

X_test_school_state_label_encoding = label.transform(X_test['school_state'].values)
```

In [30]:

```
data['school_state'].nunique()
```

Out[30]:

```
51
```

In [31]:

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(data['teacher_prefix'].values)


X_train_teacher_prefix = label.transform(X_train['teacher_prefix'].values)

X_test_teacher_prefix = label.transform(X_test['teacher_prefix'].values)
```

In [32]:

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(data['clean_categories'].values)


X_train_clean_categories = label.transform(X_train['clean_categories'].values)

X_test_clean_categories = label.transform(X_test['clean_categories'].values)
```

In [33]:

```
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
label.fit(data['project_grade_category'].values)


X_train_project_grade_category = label.transform(X_train['project_grade_category'].values)

X_test_project_grade_category = label.transform(X_test['project_grade_category'].values)
```

In [34]:

```



from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(1,-1))


X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(1,-1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(1,-1))
X_train_price_norm=X_train_price_norm.reshape(-1,1)
X_test_price_norm=X_test_price_norm.reshape(-1,1)
```

In [35]:

```



from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1
,-1))

X_train_teacher_number_of_previously_posted_projects_norm= normalizer.transform(X_train
['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X_test[
'teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

X_train_teacher_number_of_previously_posted_projects_norm=X_train_teacher_number_of_pre
viously_posted_projects_norm.reshape(-1,1)

X_test_teacher_number_of_previously_posted_projects_norm=X_test_teacher_number_of_previ
ously_posted_projects_norm.reshape(-1,1)
print("After vectorizations")
print(X_train_teacher_number_of_previously_posted_projects_norm.shape, Y_train.shape)
#print(X_cv_teacher_number_of_previously_posted_projects_norm.shape, y_cv.shape)
print(X_test_teacher_number_of_previously_posted_projects_norm.shape, Y_test.shape)
print("="*100)
```

```
After vectorizations
(73196, 1) (73196,)
(36052, 1) (36052,)
==========================================================================
==========================
```

In [36]:

```
X_train_numeric = np.concatenate((X_train_price_norm , X_train_teacher_number_of_previo
usly_posted_projects_norm) , axis = 1)

X_test_numeric= np.concatenate((X_test_price_norm , X_test_teacher_number_of_previously
_posted_projects_norm) , axis = 1)
```

In [37]:

```
X_train_numeric.shape
```

Out[37]:

```
(73196, 2)
```

In [38]:

```
data['clean_subcategories'].nunique()
```

Out[38]:

```
401
```

# Reference:https://stackoverflow.com/questions/57574501/how-to-use-sklearn-auc-in-tensorflow-keras-model-metrics (https://stackoverflow.com/questions/57574501/how-to-use-sklearn-auc-in-tensorflow-keras-model-metrics)

In [39]:

```
def auc1(y_true, y_pred):
    if len(np.unique(y_true[:,1])) == 1:
        return 0.5
    else:
        return roc_auc_score(y_true, y_pred)

def auroc(y_true, y_pred):
    return tf.py_function(auc1, (y_true, y_pred), tf.double)
```

In [40]:

```
from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras import regularizers
from keras.regularizers import l2
from keras.layers import Flatten
from keras.layers import Dense, Input , Dropout
from keras.layers import concatenate
from keras.layers.normalization import BatchNormalization
from keras.callbacks import TensorBoard
```

In [41]:

```
from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras import regularizers
from keras.regularizers import l2
from keras.layers import Flatten
from keras.layers import Dense, Input , Dropout
from keras.layers import concatenate
from keras.layers.normalization import BatchNormalization
from keras.callbacks import TensorBoard


essay = Input(shape=(250,), name='essay_input')

X = Embedding(output_dim=300, input_dim=number_of_words_in_corpus+1, input_length=250 , weights=[embedding_matrix],trainable=False)(essay)
lstm_essay = LSTM(200,recurrent_dropout=0.5,return_sequences=True)(X)
flatten_1 = Flatten()(lstm_essay)



school_state = Input(shape=(1,), name='school_state')
X_school_state = Embedding(output_dim=int(np.sqrt(data['school_state'].nunique())) , input_dim=data['school_state'].nunique(), input_length=1)(school_state)
flatten_2 = Flatten()(X_school_state)



teacher_prefix = Input(shape=(1,), name='teacher_prefix')
X_teacher_prefix = Embedding(output_dim=int(np.sqrt(data['teacher_prefix'].nunique())) , input_dim=data['teacher_prefix'].nunique(), input_length=1)(teacher_prefix)
flatten_3 = Flatten()(X_teacher_prefix)



clean_categories = Input(shape=(1,), name='clean_categories')
X_clean_categories = Embedding(output_dim=int(np.sqrt(data['clean_categories'].nunique())), input_dim=data['clean_categories'].nunique(), input_length=1)(clean_categories)
flatten_4 = Flatten()(X_clean_categories)



clean_subcategories = Input(shape=(3,), name='clean_subcategories')
X_clean_subcategories = Embedding(output_dim=int(np.sqrt(data['clean_subcategories'].nunique())), input_dim=data['clean_subcategories'].nunique(), input_length=3)(clean_subcategories)
flatten_5 = Flatten()(X_clean_subcategories)




project_grade_category = Input(shape=(1,), name='project_grade_category')
X_project_grade_category = Embedding(output_dim=int(np.sqrt(data['project_grade_category'].nunique())), input_dim=data['project_grade_category'].nunique(), input_length=1)(project_grade_category)
flatten_6 = Flatten()(X_project_grade_category)
```

```
numeric_features = Input(shape=(2,) , name="numerical_features")
numeric_dense = Dense(128, activation='relu' , kernel_initializer='he_normal')(numeric_
features)


X_concat = concatenate([flatten_1 , flatten_2 , flatten_3 ,flatten_4 , flatten_5 , flat
ten_6 , numeric_dense])
model = Dense(300, activation="relu", kernel_initializer="he_normal" ,kernel_regularize
r=regularizers.l2(0.001))(X_concat)

model = Dropout(0.5)(model)

model = Dense(200,activation="relu",kernel_initializer="glorot_normal")(model)

model = BatchNormalization()(model)

model = Dropout(0.5)(model)

model = Dense(80,activation="relu", kernel_initializer="glorot_normal" )(model)


output = Dense(2, activation='softmax', name='output')(model)

model_1 = Model(inputs=[essay, school_state ,teacher_prefix,clean_categories,
                       clean_subcategories ,project_grade_category ,numeric_features ],
outputs=[output])



print(model_1.summary())
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernel since it doesn't m
eet the cuDNN kernel criteria. It will use generic GPU kernel as fallback
when running on GPU
Model: "functional_1"
__________________________________________________________________________
________________________
Layer (type)                    Output Shape         Param #     Connected
to
==========================================================================
========================
essay_input (InputLayer)        [(None, 250)]        0
__________________________________________________________________________
________________________
embedding (Embedding)           (None, 250, 300)     14439000    essay_inp
ut[0][0]
__________________________________________________________________________
________________________
school_state (InputLayer)       [(None, 1)]          0
__________________________________________________________________________
________________________
teacher_prefix (InputLayer)     [(None, 1)]          0
__________________________________________________________________________
________________________
clean_categories (InputLayer)   [(None, 1)]          0
__________________________________________________________________________
________________________
clean_subcategories (InputLayer [(None, 3)]          0
__________________________________________________________________________
________________________
project_grade_category (InputLa [(None, 1)]          0
__________________________________________________________________________
________________________
lstm (LSTM)                     (None, 250, 200)     400800      embedding
[0][0]
__________________________________________________________________________
________________________
embedding_1 (Embedding)         (None, 1, 7)         357         school_st
ate[0][0]
__________________________________________________________________________
________________________
embedding_2 (Embedding)         (None, 1, 2)         10          teacher_p
refix[0][0]
__________________________________________________________________________
________________________
embedding_3 (Embedding)         (None, 1, 7)         357         clean_cat
egories[0][0]
__________________________________________________________________________
________________________
embedding_4 (Embedding)         (None, 3, 20)        8020        clean_sub
categories[0][0]
__________________________________________________________________________
________________________
embedding_5 (Embedding)         (None, 1, 2)         8           project_g
rade_category[0][0]
__________________________________________________________________________
________________________
numerical_features (InputLayer) [(None, 2)]          0
__________________________________________________________________________
________________________
flatten (Flatten)               (None, 50000)        0           lstm[0]
[0]
```

```
__________________________________________________________________________

________________________
flatten_1 (Flatten)             (None, 7)            0           embedding
_1[0][0]

__________________________________________________________________________

________________________
flatten_2 (Flatten)             (None, 2)            0           embedding
_2[0][0]

__________________________________________________________________________

________________________
flatten_3 (Flatten)             (None, 7)            0           embedding
_3[0][0]

__________________________________________________________________________

________________________
flatten_4 (Flatten)             (None, 60)           0           embedding
_4[0][0]

__________________________________________________________________________

________________________
flatten_5 (Flatten)             (None, 2)            0           embedding
_5[0][0]

__________________________________________________________________________

________________________
dense (Dense)                   (None, 128)          384         numerical
_features[0][0]

__________________________________________________________________________

________________________
concatenate (Concatenate)       (None, 50206)        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, 300)          15062100    concatena
te[0][0]

__________________________________________________________________________

________________________
dropout (Dropout)               (None, 300)          0           dense_1
[0][0]

__________________________________________________________________________

________________________
dense_2 (Dense)                 (None, 200)          60200       dropout
[0][0]

__________________________________________________________________________

________________________
batch_normalization (BatchNorma (None, 200)          800         dense_2
[0][0]

__________________________________________________________________________

________________________
dropout_1 (Dropout)             (None, 200)          0           batch_nor
malization[0][0]

__________________________________________________________________________
```

```
________________________
dense_3 (Dense)                 (None, 80)           16080       dropout_1
[0][0]
__________________________________________________________________________
________________________
output (Dense)                  (None, 2)            162         dense_3
[0][0]
==========================================================================
========================
Total params: 29,988,278
Trainable params: 15,548,878
Non-trainable params: 14,439,400
__________________________________________________________________________
________________________
None
```

In [42]:

```
train = [X_train_pad,X_train_school_state_label_encoding.reshape(-1,1),X_train_teacher_prefix.reshape(-1,1),X_train_clean_categories.reshape(-1,1),np.array(X_train['clean_subcategories_label_'].to_list()),X_train_project_grade_category.reshape(-1,1),X_train_numeric]

test = [X_test_pad,X_test_school_state_label_encoding.reshape(-1,1),X_test_teacher_prefix.reshape(-1,1),X_test_clean_categories.reshape(-1,1),np.array(X_test['clean_subcategories_label_'].to_list()),X_test_project_grade_category.reshape(-1,1),X_test_numeric]
```

In [43]:

```
X_train_pad.shape
```

Out[43]:

```
(73196, 250)
```

In [44]:

```
X_train_school_state_label_encoding.reshape(-1,1).shape
```

Out[44]:

```
(73196, 1)
```

In [45]:

```
X_train_teacher_prefix.reshape(-1,1).shape
```

Out[45]:

```
(73196, 1)
```

In [46]:

```
np.array(X_train['clean_subcategories_label_'].to_list()).shape
```

Out[46]:

```
(73196, 3)
```

In [47]:

```
from keras.utils import np_utils

y_train = np_utils.to_categorical(Y_train, 2)
y_test = np_utils.to_categorical(Y_test, 2)
```

In [48]:

```
y_train
```

Out[48]:

```
array([[0., 1.],
       [1., 0.],
       [0., 1.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

In [49]:

```
import numpy as np
import tensorflow as tf
from keras.callbacks import Callback
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score,r
oc_auc_score
class Metrics(tf.keras.callbacks.Callback):
  def __init__(self, validation_data=None,validation_target=None):
    super(Metrics, self).__init__()
    self.validation_data = validation_data
    self.validation_target=validation_target
    # best_weights to store the weights at which the minimum loss occurs.

  def on_train_begin(self, logs={}):
    self.val_f1s=[]
    self.val_auc=[]

  def on_epoch_end(self, epoch, logs={}):


    val_predict =np.array((self.model.predict(self.validation_data)))
    val_targ = np.array(self.validation_target,dtype=int)
    #_val_f1 = f1_score(val_targ, val_predict)
    _val_auc=roc_auc_score(val_targ[:,0], val_predict[:,0],average='macro')
   # self.val_f1s.append(_val_f1)
    #self.val_auc.append(_val_auc)
    print(' -val_auc_score: '+str(_val_auc))
```

In [50]:

```
#Input layer
import warnings
warnings.filterwarnings("ignore")
import datetime
import os
checkpoint1 = ModelCheckpoint("model_1.h5",

                             monitor="val_loss",
                             mode="auto",
                             save_best_only = True,
                             verbose=1)
earlystop1 = EarlyStopping(monitor = 'val_loss',
                            mode="auto",
                            min_delta = 0,
                            patience = 4,
                            verbose = 2)

log_dir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,
write_graph=True,write_grads=True)

metric=Metrics(test, y_test)

callbacks_1= [checkpoint1,earlystop1,tensorboard_callback]
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
```

In [51]:

```
model_1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[auroc])
history1 = model_1.fit(train, y_train, batch_size=512, epochs=10, verbose=1,callbacks=callbacks_1, validation_data=(test, y_test))
```

```
Epoch 1/10
  1/143 [..............................] - ETA: 0s - loss: 2.5252 - auroc:
0.4815WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tenso
rflow/python/ops/summary_ops_v2.py:1277: stop (from tensorflow.python.eage
r.profiler) is deprecated and will be removed after 2020-07-01.
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
  2/143 [..............................] - ETA: 5:51 - loss: 2.4828 - auro
c: 0.4741WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow
compared to the batch time (batch time: 1.3537s vs `on_train_batch_end` ti
me: 3.6372s). Check your callbacks.
143/143 [==============================] - ETA: 0s - loss: 0.9859 - auroc:
0.5491
Epoch 00001: val_loss improved from inf to 0.59027, saving model to model_
1.h5
143/143 [==============================] - 142s 991ms/step - loss: 0.9859
- auroc: 0.5491 - val_loss: 0.5903 - val_auroc: 0.6661
Epoch 2/10
143/143 [==============================] - ETA: 0s - loss: 0.5503 - auroc:
0.6541
Epoch 00002: val_loss improved from 0.59027 to 0.49614, saving model to mo
del_1.h5
143/143 [==============================] - 135s 942ms/step - loss: 0.5503
- auroc: 0.6541 - val_loss: 0.4961 - val_auroc: 0.7167
Epoch 3/10
143/143 [==============================] - ETA: 0s - loss: 0.4909 - auroc:
0.7004
Epoch 00003: val_loss improved from 0.49614 to 0.47218, saving model to mo
del_1.h5
143/143 [==============================] - 131s 917ms/step - loss: 0.4909
- auroc: 0.7004 - val_loss: 0.4722 - val_auroc: 0.7304
Epoch 4/10
143/143 [==============================] - ETA: 0s - loss: 0.4617 - auroc:
0.7235
Epoch 00004: val_loss improved from 0.47218 to 0.44821, saving model to mo
del_1.h5
143/143 [==============================] - 137s 959ms/step - loss: 0.4617
- auroc: 0.7235 - val_loss: 0.4482 - val_auroc: 0.7408
Epoch 5/10
143/143 [==============================] - ETA: 0s - loss: 0.4447 - auroc:
0.7353
Epoch 00005: val_loss improved from 0.44821 to 0.44606, saving model to mo
del_1.h5
143/143 [==============================] - 140s 976ms/step - loss: 0.4447
- auroc: 0.7353 - val_loss: 0.4461 - val_auroc: 0.7387
Epoch 6/10
143/143 [==============================] - ETA: 0s - loss: 0.4397 - auroc:
0.7462
Epoch 00006: val_loss improved from 0.44606 to 0.44260, saving model to mo
del_1.h5
143/143 [==============================] - 142s 992ms/step - loss: 0.4397
- auroc: 0.7462 - val_loss: 0.4426 - val_auroc: 0.7496
Epoch 7/10
143/143 [==============================] - ETA: 0s - loss: 0.4410 - auroc:
0.7548
Epoch 00007: val_loss did not improve from 0.44260
143/143 [==============================] - 138s 965ms/step - loss: 0.4410
- auroc: 0.7548 - val_loss: 0.4605 - val_auroc: 0.7530
Epoch 8/10
143/143 [==============================] - ETA: 0s - loss: 0.4365 - auroc:
0.7592
```

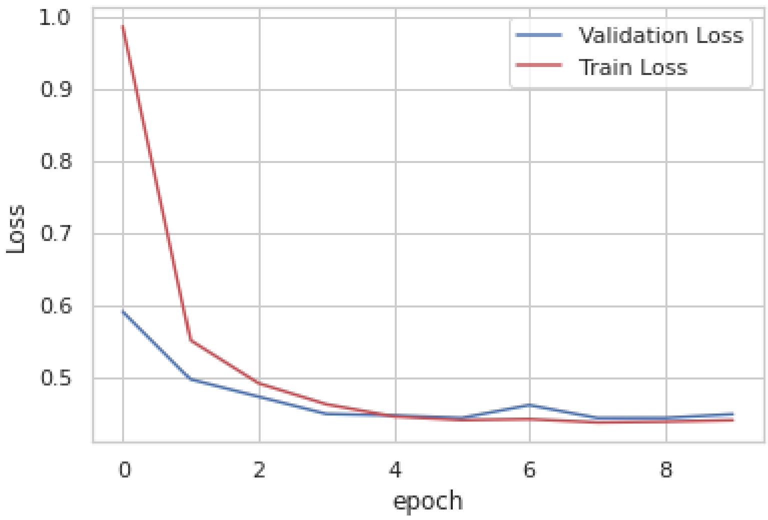
```
Epoch 00008: val_loss improved from 0.44260 to 0.44257, saving model to mo
del_1.h5
143/143 [==============================] - 140s 981ms/step - loss: 0.4365
- auroc: 0.7592 - val_loss: 0.4426 - val_auroc: 0.7530
Epoch 9/10
143/143 [==============================] - ETA: 0s - loss: 0.4375 - auroc:
0.7719
Epoch 00009: val_loss did not improve from 0.44257
143/143 [==============================] - 138s 967ms/step - loss: 0.4375
- auroc: 0.7719 - val_loss: 0.4427 - val_auroc: 0.7512
Epoch 10/10
143/143 [==============================] - ETA: 0s - loss: 0.4395 - auroc:
0.7748
Epoch 00010: val_loss did not improve from 0.44257
143/143 [==============================] - 138s 968ms/step - loss: 0.4395
- auroc: 0.7748 - val_loss: 0.4479 - val_auroc: 0.7531
```

In [52]:

```
%matplotlib inline
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Loss')
x = list(range(0,10))
vy = history1.history['val_loss']
ty = history1.history['loss']
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
```

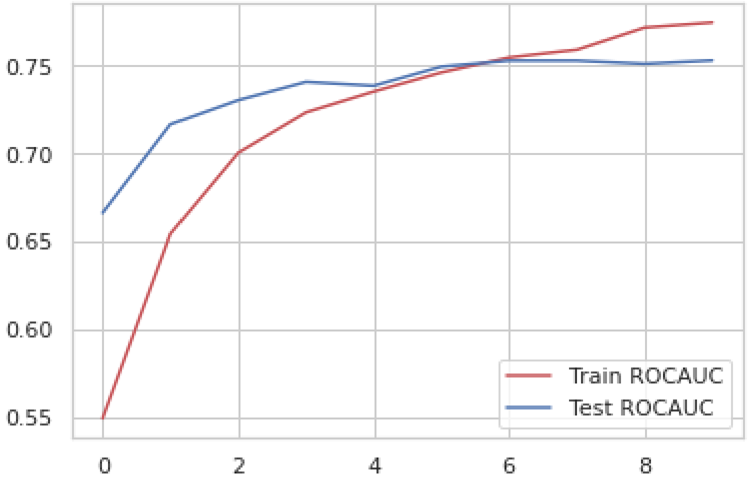
Out[52]:

```
<matplotlib.legend.Legend at 0x7f09f688fbe0>
```



In [53]:

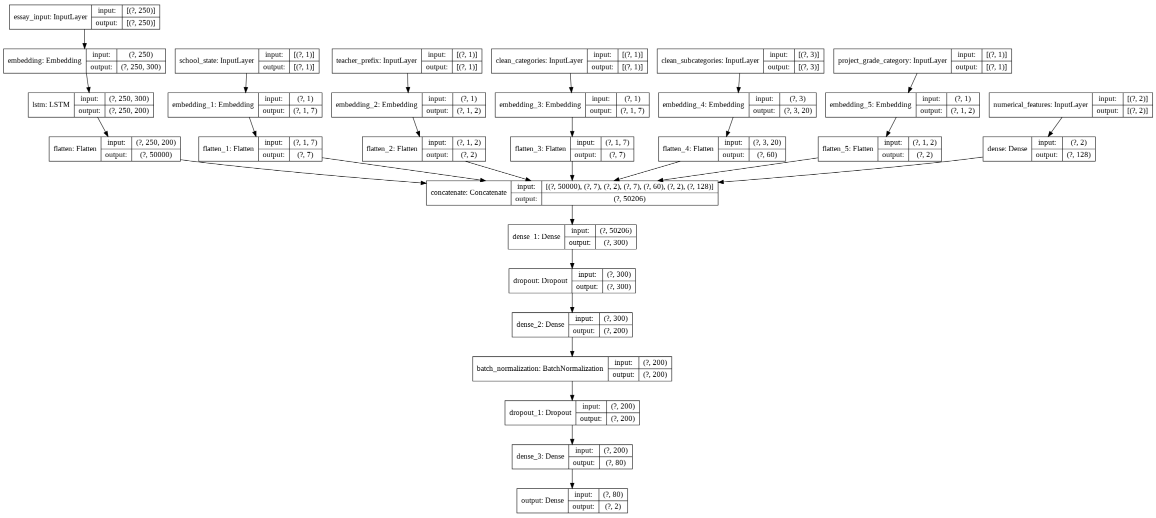
```
plt.plot(history1.history['auroc'], 'r')
plt.plot(history1.history['val_auroc'], 'b')
plt.legend({'Train ROCAUC': 'r', 'Test ROCAUC':'b'})
plt.show()
```



In [54]:

```
dot_img_file = '/tmp/model_1.png'
tf.keras.utils.plot_model(model_1, to_file=dot_img_file, show_shapes=True)
```

Out[54]:


essay_input: InputLayer input: [(?, 250)] output: [(?, 250)]
embedding: Embedding input: (?, 250) output: (?, 250, 300)
school_state: InputLayer input: [(?, 1)] output: [(?, 1)]
teacher_prefix: InputLayer input: [(?, 1)] output: [(?, 1)]
clean_categories: InputLayer input: [(?, 1)] output: [(?, 1)]
clean_subcategories: InputLayer input: [(?, 3)] output: [(?, 3)]
project_grade_category: InputLayer input: [(?, 1)] output: [(?, 1)]
lstm: LSTM input: (?, 250, 300) output: (?, 250, 200)
embedding_1: Embedding input: (?, 1) output: (?, 1, 7)
embedding_2: Embedding input: (?, 1) output: (?, 1, 2)
embedding_3: Embedding input: (?, 1) output: (?, 1, 7)
embedding_4: Embedding input: (?, 3) output: (?, 3, 20)
embedding_5: Embedding input: (?, 1) output: (?, 1, 2)
numerical_features: InputLayer input: [(?, 2)] output: [(?, 2)]
flatten: Flatten input: (?, 250, 200) output: (?, 50000)
flatten_1: Flatten input: (?, 1, 7) output: (?, 7)
flatten_2: Flatten input: (?, 1, 2) output: (?, 2)
flatten_3: Flatten input: (?, 1, 7) output: (?, 7)
flatten_4: Flatten input: (?, 3, 20) output: (?, 60)
flatten_5: Flatten input: (?, 1, 2) output: (?, 2)
dense: Dense input: (?, 2) output: (?, 128)
concatenate: Concatenate input: [(?, 50000), (?, 7), (?, 2), (?, 7), (?, 60), (?, 2), (?, 128)] output: (?, 50206)
dense_1: Dense input: (?, 50206) output: (?, 300)
dropout: Dropout input: (?, 300) output: (?, 300)
dense_2: Dense input: (?, 300) output: (?, 200)
batch_normalization: BatchNormalization input: (?, 200) output: (?, 200)
dropout_1: Dropout input: (?, 200) output: (?, 200)
dense_3: Dense input: (?, 200) output: (?, 80)
output: Dense input: (?, 80) output: (?, 2)

# Model_2

## 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.

```
    1. Train the TF-IDF on the Train data

    2. Get the idf value for each word we have in the train data.

    3. Remove the low idf value and high idf value words from our data. Do some anal
    ysis on the Idf values and based on those values choose the low and high thresho
    ld value. Because very frequent words and very very rare words don't give much i
    nformation. (you can plot a box plots and take only the idf scores within IQR ra
    nge and corresponding words)

    4. Train the LSTM after removing the Low and High idf value words. (In model-1 T
    rain on total data but in Model-2 train on data after removing some words based
     on IDF values)
```

In [55]:

```
vectorizer = TfidfVectorizer()
vectorizer.fit_transform(X_train["essay"])
```

Out[55]:

```
<73196x48093 sparse matrix of type '<class 'numpy.float64'>'
        with 7908625 stored elements in Compressed Sparse Row format>
```

## Depends upon idf percentile value, we are going to take specific words

In [56]:

```
arr=vectorizer.idf_
print("10th percentile of arr : ",
       np.percentile(arr, 10))
print("40th percentile of arr : ",
       np.percentile(arr, 40))
print("50th percentile of arr : ",
       np.percentile(arr, 50))
print("25th percentile of arr : ",
       np.percentile(arr, 25))
print("75th percentile of arr : ",
       np.percentile(arr, 75))
print("90th percentile of arr : ",
       np.percentile(arr, 90))
print("95th percentile of arr : ",
       np.percentile(arr, 95))
print("98th percentile of arr : ",
       np.percentile(arr, 98))
print("99th percentile of arr : ",
       np.percentile(arr, 99))
```

```
10th percentile of arr :  7.473521896790654
40th percentile of arr :  10.591471803068893
50th percentile of arr :  11.102297426834886
25th percentile of arr :  9.492859514400784
75th percentile of arr :  11.50776253494305
90th percentile of arr :  11.507762534943051
95th percentile of arr :  11.50776253494305
98th percentile of arr :  11.50776253494305
99th percentile of arr :  11.50776253494305
```

In [57]:

```
mid_idf=[str(i) for i,j in zip(vectorizer.get_feature_names() ,vectorizer.idf_) if j>2
and j<11.5]
```

In [58]:

```
len(set(mid_idf))
```

Out[58]:

```
29389
```

In [59]:

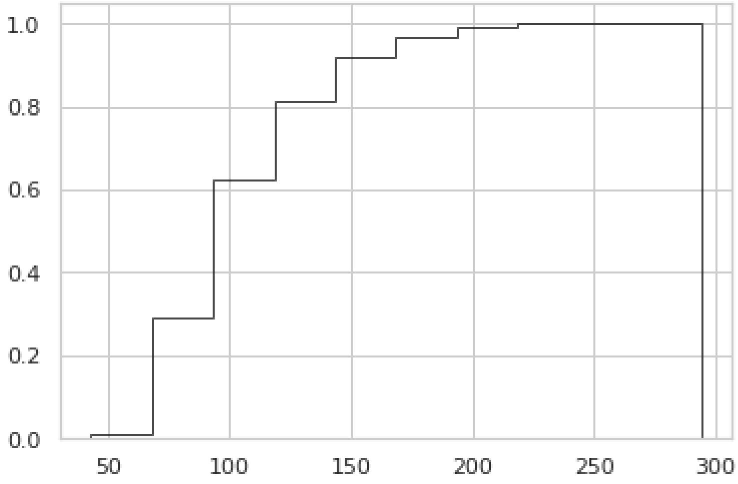
```
tokenizer = Tokenizer(num_words= 100000 )
tokenizer.fit_on_texts(mid_idf)

X_train['essay_tok_mid']  = tokenizer.texts_to_sequences(X_train['essay'].values)

X_test['essay_tok_mid']   = tokenizer.texts_to_sequences(X_test['essay'].values)
```

In [60]:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_theme(style="whitegrid")
plt.hist(X_train['essay_tok_mid'].apply(lambda x: len(x)),cumulative=True, density=True
,label='CDF', alpha=0.8, color='k',histtype='step')
#np.max(X_train['es_tok'].apply(lambda x: len(x)))
```

Out[60]:

```
(array([0.00885294, 0.29154599, 0.62286191, 0.8147713 , 0.91685338,
        0.96917864, 0.99147494, 0.99866113, 0.99986338, 1.        ]),
 array([ 43. ,  68.1,  93.2, 118.3, 143.4, 168.5, 193.6, 218.7, 243.8,
        268.9, 294. ]),
 <a list of 1 Patch objects>)
```



In [101]:

```
from keras.preprocessing import sequence
max_review_length = 200
X_train_pad1 = sequence.pad_sequences(X_train['essay_tok_mid'].values, maxlen=max_revie
w_length)

X_test_pad1  = sequence.pad_sequences(X_test['essay_tok_mid'].values, maxlen=max_review
_length)
```

In [62]:

```
X_train_pad1.shape
```

Out[62]:

```
(73196, 200)
```

In [117]:

```
from keras.models import Model
from keras.layers import Input
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras import regularizers
from keras.regularizers import l2
from keras.layers import Flatten
from keras.layers import Dense, Input , Dropout
from keras.layers import concatenate
from keras.layers.normalization import BatchNormalization
from keras.callbacks import TensorBoard


essay1 = Input(shape=(200,), name='essay_input1')

X = Embedding(len(mid_idf)+1,300,input_length=200,trainable=True)(essay1)
lstm_essay = LSTM(100,recurrent_dropout=0.5,return_sequences=True,kernel_regularizer=regularizers.l2(0.001))(X)
flatten_1_mid = Flatten()(lstm_essay)




school_state1 = Input(shape=(1,), name='school_state1')
X_school_state = Embedding(output_dim=int(np.sqrt(data['school_state'].nunique())) , input_dim=data['school_state'].nunique(), input_length=1)(school_state1)
flatten_2 = Flatten()(X_school_state)




teacher_prefix1 = Input(shape=(1,), name='teacher_prefix1')
X_teacher_prefix = Embedding(output_dim=int(np.sqrt(data['teacher_prefix'].nunique())), input_dim=data['teacher_prefix'].nunique(), input_length=1)(teacher_prefix1)
flatten_3 = Flatten()(X_teacher_prefix)




clean_categories1 = Input(shape=(1,), name='clean_categories1')
X_clean_categories = Embedding(output_dim=int(np.sqrt(data['clean_categories'].nunique())), input_dim=data['clean_categories'].nunique(), input_length=1)(clean_categories1)
flatten_4 = Flatten()(X_clean_categories)




clean_subcategories1 = Input(shape=(3,), name='clean_subcategories1')
X_clean_subcategories = Embedding(output_dim=int(np.sqrt(data['clean_subcategories'].nunique())), input_dim=data['clean_subcategories'].nunique(), input_length=3)(clean_subcategories1)
flatten_5 = Flatten()(X_clean_subcategories)





project_grade_category1 = Input(shape=(1,), name='project_grade_category1')
X_project_grade_category = Embedding(output_dim=int(np.sqrt(data['project_grade_category'].nunique())), input_dim=data['project_grade_category'].nunique(), input_length=1)(project_grade_category1)
flatten_6 = Flatten()(X_project_grade_category)
```

```
numeric_features1 = Input(shape=(2,) , name="numerical_features1")
numeric_dense = Dense(128, activation='relu' , kernel_initializer='he_normal')(numeric_
features1)


X_concat = concatenate([flatten_1_mid , flatten_2 , flatten_3 ,flatten_4 , flatten_5 ,
flatten_6 , numeric_dense])
model = Dense(50, activation="relu", kernel_initializer="he_normal")(X_concat)

model = Dropout(0.5)(model)

model = Dense(100,activation="relu",kernel_initializer="glorot_normal")(model)

model = BatchNormalization()(model)

model = Dropout(0.5)(model)

model = Dense(80,activation="sigmoid", kernel_initializer="glorot_normal")(model)


output1 = Dense(2, activation='softmax', name='output1')(model)

model_2 = Model(inputs=[essay1, school_state1 ,teacher_prefix1,clean_categories1,
                       clean_subcategories1 ,project_grade_category1 ,numeric_features1
],outputs=[output1])



print(model_2.summary())
```

```
WARNING:tensorflow:Layer lstm_12 will not use cuDNN kernel since it does
n't meet the cuDNN kernel criteria. It will use generic GPU kernel as fa
llback when running on GPU
Model: "functional_25"
________________________________________________________________________
__________________________
Layer (type)                    Output Shape         Param #     Connect
ed to
========================================================================
==========================
essay_input1 (InputLayer)       [(None, 200)]        0
________________________________________________________________________
__________________________
embedding_57 (Embedding)        (None, 200, 300)     8817000     essay_i
nput1[0][0]
________________________________________________________________________
__________________________
school_state1 (InputLayer)      [(None, 1)]          0
________________________________________________________________________
__________________________
teacher_prefix1 (InputLayer)    [(None, 1)]          0
________________________________________________________________________
__________________________
clean_categories1 (InputLayer)  [(None, 1)]          0
________________________________________________________________________
__________________________
clean_subcategories1 (InputLaye [(None, 3)]          0
________________________________________________________________________
__________________________
project_grade_category1 (InputL [(None, 1)]          0
________________________________________________________________________
__________________________
lstm_12 (LSTM)                  (None, 200, 100)     160400      embeddi
ng_57[0][0]
________________________________________________________________________
__________________________
embedding_58 (Embedding)        (None, 1, 7)         357         school_
state1[0][0]
________________________________________________________________________
__________________________
embedding_59 (Embedding)        (None, 1, 2)         10          teacher
_prefix1[0][0]
________________________________________________________________________
__________________________
embedding_60 (Embedding)        (None, 1, 7)         357         clean_c
ategories1[0][0]
________________________________________________________________________
__________________________
embedding_61 (Embedding)        (None, 3, 20)        8020        clean_s
ubcategories1[0][0]
________________________________________________________________________
__________________________
embedding_62 (Embedding)        (None, 1, 2)         8           project
_grade_category1[0][0]
________________________________________________________________________
__________________________
numerical_features1 (InputLayer [(None, 2)]          0
________________________________________________________________________
__________________________
flatten_60 (Flatten)            (None, 20000)        0           lstm_12
[0][0]
```

```
________________________________________________________________________
__________________________
flatten_61 (Flatten)           (None, 7)           0           embeddi
ng_58[0][0]
________________________________________________________________________
__________________________
flatten_62 (Flatten)           (None, 2)           0           embeddi
ng_59[0][0]
________________________________________________________________________
__________________________
flatten_63 (Flatten)           (None, 7)           0           embeddi
ng_60[0][0]
________________________________________________________________________
__________________________
flatten_64 (Flatten)           (None, 60)          0           embeddi
ng_61[0][0]
________________________________________________________________________
__________________________
flatten_65 (Flatten)           (None, 2)           0           embeddi
ng_62[0][0]
________________________________________________________________________
__________________________
dense_45 (Dense)               (None, 128)         384         numeric
al_features1[0][0]
________________________________________________________________________
__________________________
concatenate_12 (Concatenate)   (None, 20206)       0           flatten
_60[0][0]
                                                               flatten
_61[0][0]
                                                               flatten
_62[0][0]
                                                               flatten
_63[0][0]
                                                               flatten
_64[0][0]
                                                               flatten
_65[0][0]
                                                               dense_4
5[0][0]
________________________________________________________________________
__________________________
dense_46 (Dense)               (None, 50)          1010350     concate
nate_12[0][0]
________________________________________________________________________
__________________________
dropout_24 (Dropout)           (None, 50)          0           dense_4
6[0][0]
________________________________________________________________________
__________________________
dense_47 (Dense)               (None, 100)         5100        dropout
_24[0][0]
________________________________________________________________________
__________________________
batch_normalization_12 (BatchNo (None, 100)        400         dense_4
7[0][0]
________________________________________________________________________
__________________________
dropout_25 (Dropout)           (None, 100)         0           batch_n
ormalization_12[0][0]
________________________________________________________________________
```

```
__________________________
dense_48 (Dense)                (None, 80)           8080        dropout
_25[0][0]

________________________________________________________________________
__________________________
output1 (Dense)                 (None, 2)            162         dense_4
8[0][0]
========================================================================
==========================
Total params: 10,010,628
Trainable params: 10,010,428
Non-trainable params: 200

________________________________________________________________________
__________________________
None
```

In [103]:

```
train = [X_train_pad1,X_train_school_state_label_encoding.reshape(-1,1),X_train_teacher_prefix.reshape(-1,1),X_train_clean_categories.reshape(-1,1),np.array(X_train['clean_subcategories_label_'].to_list()),X_train_project_grade_category.reshape(-1,1),X_train_numeric]

test = [X_test_pad1,X_test_school_state_label_encoding.reshape(-1,1),X_test_teacher_prefix.reshape(-1,1),X_test_clean_categories.reshape(-1,1),np.array(X_test['clean_subcategories_label_'].to_list()),X_test_project_grade_category.reshape(-1,1),X_test_numeric]
```

In [104]:

```
from keras.utils import np_utils

y_train = np_utils.to_categorical(Y_train, 2)
y_test = np_utils.to_categorical(Y_test, 2)
```

In [66]:

```
y_test.shape
```

Out[66]:

```
(36052, 2)
```

In [66]:

```

```

In [118]:

```
#Input layer
import warnings
warnings.filterwarnings("ignore")
import datetime
import os
checkpoint1 = ModelCheckpoint("model_2.h5",

                             monitor="val_loss",
                             mode="auto",
                             save_best_only = True,
                             verbose=1)
earlystop1 = EarlyStopping(monitor = 'val_loss',
                            mode="auto",
                            min_delta = 0,
                            patience = 2,
                            verbose = 2)

log_dir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,
write_graph=True,write_grads=True)
metric=Metrics(test, y_test)


callbacks_2= [checkpoint1,earlystop1,tensorboard_callback]
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
```

In [119]:

```
model_2.compile(optimizer='Nadam', loss='categorical_crossentropy', metrics=[auroc])
history = model_2.fit(train, y_train, batch_size=600, epochs=10, verbose=1,callbacks=ca
llbacks_2, validation_data=(test, y_test))
```

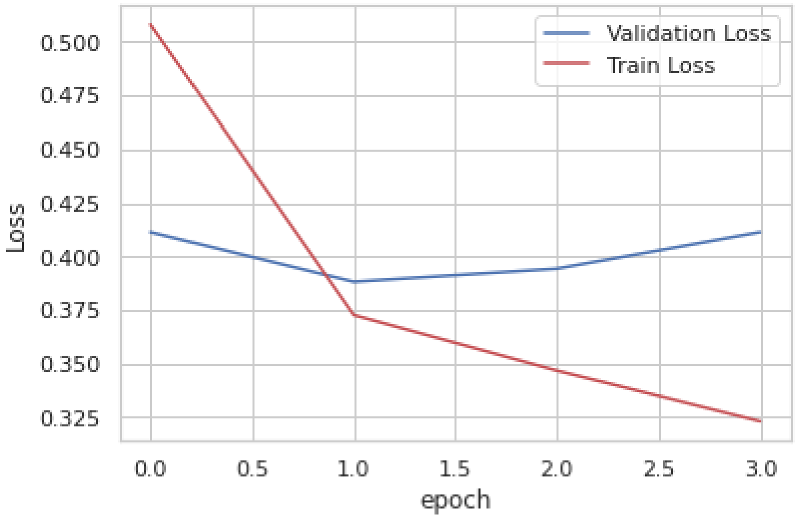
```
Epoch 1/10
  2/122 [..............................] - ETA: 4:43 - loss: 0.7612 - auro
c: 0.4918WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow
compared to the batch time (batch time: 1.3319s vs `on_train_batch_end` ti
me: 3.3977s). Check your callbacks.
122/122 [==============================] - ETA: 0s - loss: 0.5081 - auroc:
0.6630
Epoch 00001: val_loss improved from inf to 0.41136, saving model to model_
2.h5
122/122 [==============================] - 120s 982ms/step - loss: 0.5081
- auroc: 0.6630 - val_loss: 0.4114 - val_auroc: 0.7315
Epoch 2/10
122/122 [==============================] - ETA: 0s - loss: 0.3727 - auroc:
0.7649
Epoch 00002: val_loss improved from 0.41136 to 0.38823, saving model to mo
del_2.h5
122/122 [==============================] - 116s 954ms/step - loss: 0.3727
- auroc: 0.7649 - val_loss: 0.3882 - val_auroc: 0.7314
Epoch 3/10
122/122 [==============================] - ETA: 0s - loss: 0.3467 - auroc:
0.7978
Epoch 00003: val_loss did not improve from 0.38823
122/122 [==============================] - 115s 945ms/step - loss: 0.3467
- auroc: 0.7978 - val_loss: 0.3943 - val_auroc: 0.7268
Epoch 4/10
122/122 [==============================] - ETA: 0s - loss: 0.3231 - auroc:
0.8293
Epoch 00004: val_loss did not improve from 0.38823
122/122 [==============================] - 114s 938ms/step - loss: 0.3231
- auroc: 0.8293 - val_loss: 0.4114 - val_auroc: 0.7101
Epoch 00004: early stopping
```

In [120]:

```
%matplotlib inline
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Loss')
x = list(range(0,4))
vy = history.history['val_loss']
ty = history.history['loss']
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
```

Out[120]:

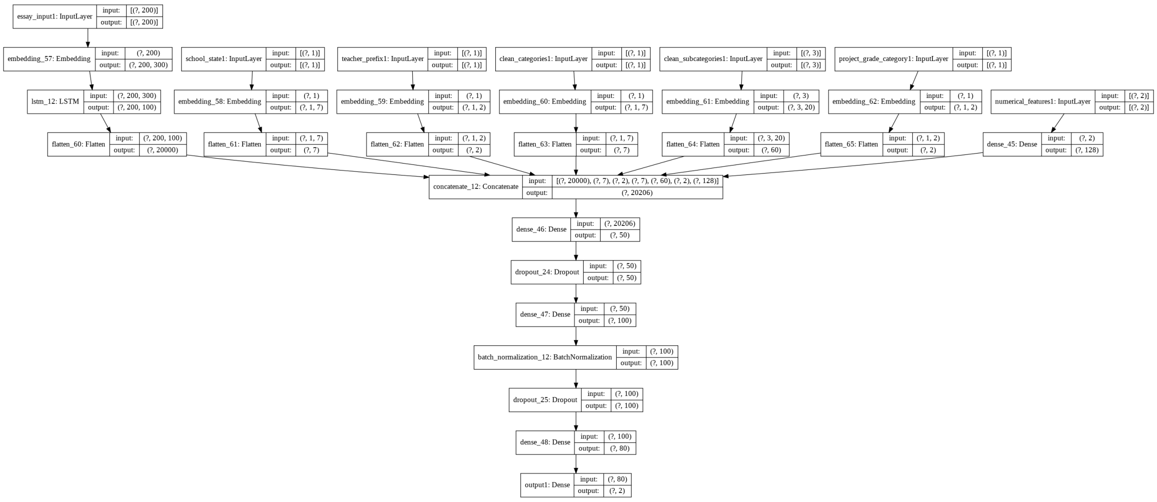
```
<matplotlib.legend.Legend at 0x7f0859e87390>
```



In [121]:

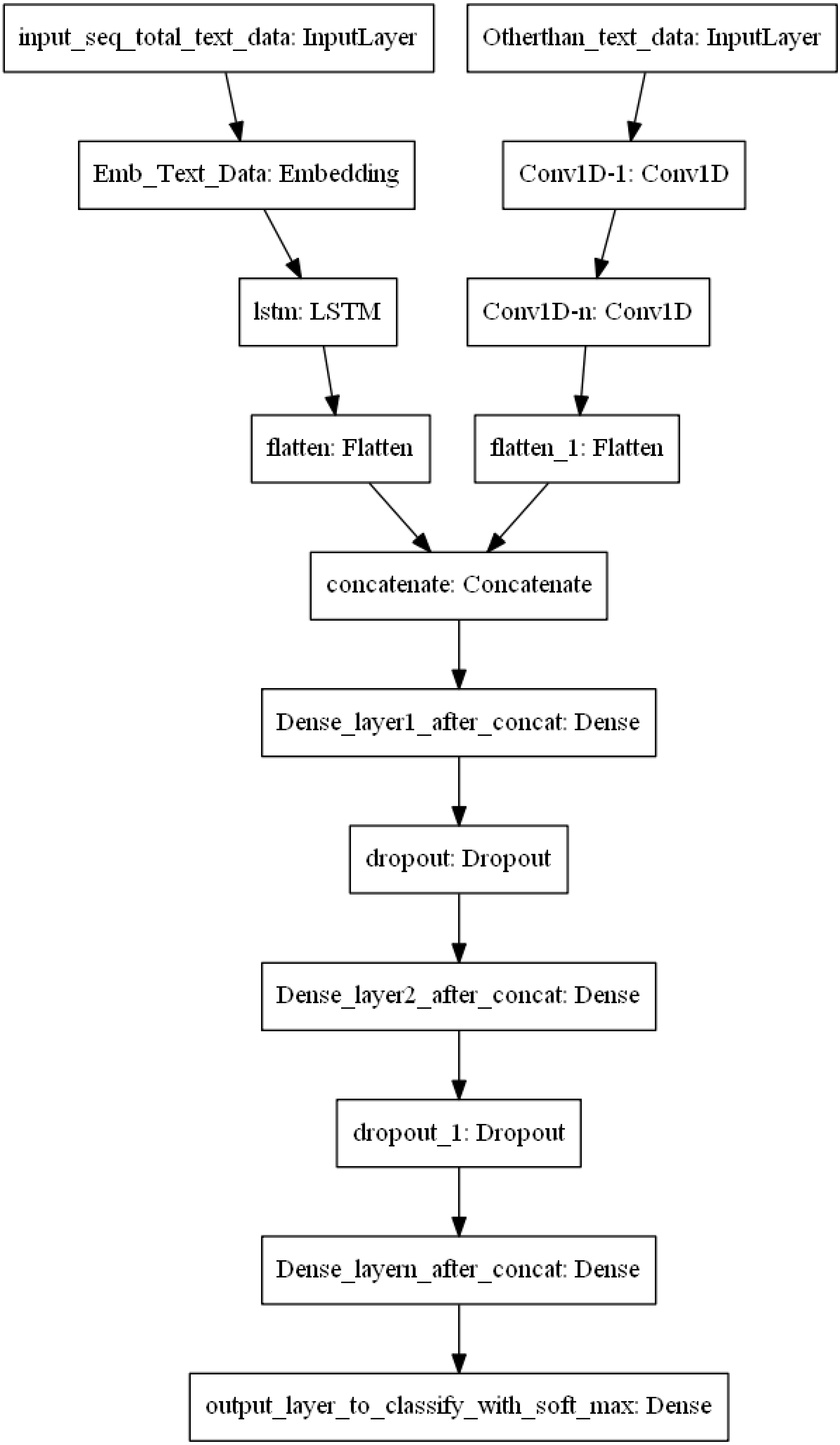
```
dot_img_file = '/tmp/model_2.png'
tf.keras.utils.plot_model(model_2, to_file=dot_img_file, show_shapes=True)
```

Out[121]:


essay_input1: InputLayer | input: [(?, 200)] | output: [(?, 200)]
embedding_57: Embedding | input: (?, 200) | output: (?, 200, 300)
school_state1: InputLayer | input: [(?, 1)] | output: [(?, 1)]
teacher_prefix1: InputLayer | input: [(?, 1)] | output: [(?, 1)]
clean_categories1: InputLayer | input: [(?, 1)] | output: [(?, 1)]
clean_subcategories1: InputLayer | input: [(?, 3)] | output: [(?, 3)]
project_grade_category1: InputLayer | input: [(?, 1)] | output: [(?, 1)]
lstm_12: LSTM | input: (?, 200, 300) | output: (?, 200, 100)
embedding_58: Embedding | input: (?, 1) | output: (?, 1, 7)
embedding_59: Embedding | input: (?, 1) | output: (?, 1, 2)
embedding_60: Embedding | input: (?, 1) | output: (?, 1, 7)
embedding_61: Embedding | input: (?, 3) | output: (?, 3, 20)
embedding_62: Embedding | input: (?, 1) | output: (?, 1, 2)
numerical_features1: InputLayer | input: [(?, 2)] | output: [(?, 2)]
flatten_60: Flatten | input: (?, 200, 100) | output: (?, 20000)
flatten_61: Flatten | input: (?, 1, 7) | output: (?, 7)
flatten_62: Flatten | input: (?, 1, 2) | output: (?, 2)
flatten_63: Flatten | input: (?, 1, 7) | output: (?, 7)
flatten_64: Flatten | input: (?, 3, 20) | output: (?, 60)
flatten_65: Flatten | input: (?, 1, 2) | output: (?, 2)
dense_45: Dense | input: (?, 2) | output: (?, 128)
concatenate_12: Concatenate | input: [(?, 20000), (?, 7), (?, 2), (?, 7), (?, 60), (?, 2), (?, 128)] | output: (?, 20206)
dense_46: Dense | input: (?, 20206) | output: (?, 50)
dropout_24: Dropout | input: (?, 50) | output: (?, 50)
dense_47: Dense | input: (?, 50) | output: (?, 100)
batch_normalization_12: BatchNormalization | input: (?, 100) | output: (?, 100)
dropout_25: Dropout | input: (?, 100) | output: (?, 100)
dense_48: Dense | input: (?, 100) | output: (?, 80)
output1: Dense | input: (?, 80) | output: (?, 2)

# Model_3

## Model-3


input_seq_total_text_data: InputLayer
Otherthan_text_data: InputLayer
Emb_Text_Data: Embedding
Conv1D-1: Conv1D
lstm: LSTM
Conv1D-n: Conv1D
flatten: Flatten
flatten_1: Flatten
concatenate: Concatenate
Dense_layer1_after_concat: Dense
dropout: Dropout
Dense_layer2_after_concat: Dense
dropout_1: Dropout
Dense_layern_after_concat: Dense
output_layer_to_classify_with_soft_max: Dense

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

- **input_seq_total_text_data**:

```
      . Use text column('essay'), and use the Embedding layer to get word vector
    s.

      . Use given predefined glove word vectors, don't train any word vectors.

      . Use LSTM that is given above, get the LSTM output and Flatten that outpu
    t.

      . You are free to preprocess the input text as you needed.
```

- **Other_than_text_data**:

```
      . Convert all your Categorical values to onehot coded and then concatenate
    all these onehot vectors

      . Neumerical values and use CNN1D (https://keras.io/getting-started/sequen
    tial-model-guide/#sequence-classification-with-1d-convolutions) as shown in
     above figure.

      . You are free to choose all CNN parameters like kernel sizes, stride.
```

</pre>

In [73]:

```python
from keras.layers import Conv1D
from sklearn.feature_extraction.text import CountVectorizer
```

In [74]:

```python
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values)
x_train_state_one_hot = vectorizer.transform(X_train['school_state'].values)
x_test_state_one_hot = vectorizer.transform(X_test['school_state'].values)



print(x_train_state_one_hot.shape, y_train.shape)
print(x_test_state_one_hot.shape, y_test.shape)
```

```
(73196, 51) (73196, 2)
(36052, 51) (36052, 2)
```

In [75]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values)

x_train_categories_one_hot = vectorizer.transform(X_train['clean_categories'].values)
x_test_categories_one_hot = vectorizer.transform(X_test['clean_categories'].values)





print(x_train_categories_one_hot.shape, y_train.shape)
print(x_test_categories_one_hot.shape, y_test.shape)
```

```
(73196, 9) (73196, 2)
(36052, 9) (36052, 2)
```

In [76]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values)

x_train_subcategories_one_hot = vectorizer.transform(X_train['clean_subcategories'].val
ues)
x_test_subcategories_one_hot = vectorizer.transform(X_test['clean_subcategories'].value
s)


print(x_train_subcategories_one_hot.shape, y_train.shape)
print(x_test_subcategories_one_hot.shape, y_test.shape)
```

```
(73196, 30) (73196, 2)
(36052, 30) (36052, 2)
```

In [77]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values)
x_train_teacher_prefix_one_hot = vectorizer.transform(X_train['teacher_prefix'].values)
x_test_teacher_prefix_one_hot = vectorizer.transform(X_test['teacher_prefix'].values)

print(x_train_teacher_prefix_one_hot.shape, y_train.shape)
print(x_test_teacher_prefix_one_hot.shape, y_test.shape)
```

```
(73196, 5) (73196, 2)
(36052, 5) (36052, 2)
```

In [78]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['project_grade_category'].values)
x_train_project_grade_one_hot = vectorizer.transform(X_train['project_grade_category'].values)
x_test_project_grade_one_hot = vectorizer.transform(X_test['project_grade_category'].values)

print(x_train_project_grade_one_hot.shape, y_train.shape)
print(x_test_project_grade_one_hot.shape, y_test.shape)
```

```
(73196, 4) (73196, 2)
(36052, 4) (36052, 2)
```

In [79]:

```
from scipy.sparse import hstack
```

In [80]:

```
train_features_wot= hstack((x_train_project_grade_one_hot,x_train_teacher_prefix_one_hot,x_train_categories_one_hot,x_train_subcategories_one_hot,x_train_state_one_hot,X_train_price_norm,X_train_teacher_number_of_previously_posted_projects_norm)).todense()
test_features_wot = hstack((x_test_project_grade_one_hot,x_test_teacher_prefix_one_hot,x_test_categories_one_hot,x_test_subcategories_one_hot,x_test_state_one_hot,X_test_price_norm,X_test_teacher_number_of_previously_posted_projects_norm)).todense()
```

In [81]:

```
rest_train = np.expand_dims(train_features_wot,2)
rest_test = np.expand_dims(test_features_wot,2)
```

In [82]:

```
print("train data shape",rest_train.shape)
print("test data shape",rest_test.shape)
```

```
train data shape (73196, 101, 1)
test data shape (36052, 101, 1)
```

In [83]:

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words= 1000000 )
tokenizer.fit_on_texts(X_train["essay"])
```

In [84]:

```
X_train['es_tok']  = tokenizer.texts_to_sequences(X_train['essay'].values)

X_test['es_tok']   = tokenizer.texts_to_sequences(X_test['essay'].values)
```

In [85]:

```
l=tokenizer.word_index.items()
number_of_words_in_corpus = len(tokenizer.word_index)

embedding_matrix = np.zeros((number_of_words_in_corpus+1, 300))
for word, i in l:


  if word in glove_words:
    embedding_matrix[i] =glove[word]
```

In [86]:

```
from keras.preprocessing import sequence
max_review_length = 250
X_train_pad = sequence.pad_sequences(X_train['es_tok'].values, maxlen=max_review_length
)

X_test_pad  = sequence.pad_sequences(X_test['es_tok'].values, maxlen=max_review_length)
```

In [86]:

In [93]:

```
from keras.layers import Conv1D
from keras.initializers import he_normal



essay = Input(shape=(250,))

X = Embedding(output_dim=300, input_dim=number_of_words_in_corpus+1, input_length=300 , weights=[embedding_matrix],trainable=False)(essay)
lstm_essay = LSTM(200,recurrent_dropout=0.5,return_sequences=True)(X)
flatten_1 = Flatten()(lstm_essay)





input_wot  = Input(shape=(101,1))

con = Conv1D(300 , 3 , activation='relu' ,  kernel_initializer=he_normal(seed=10) , padding='valid')(input_wot)

convo = Conv1D(150 , 3 , activation='relu' ,  kernel_initializer=he_normal(seed=0) , padding='valid')(con)

flatten_2 = Flatten()(convo)



x_concat = concatenate([flatten_1  , flatten_2])

x = Dense(120, activation="relu", kernel_initializer="he_normal",kernel_regularizer=regularizers.l2(0.001) )(x_concat)

x=Dropout(0.5)(x)

x = Dense(200,activation="sigmoid",kernel_initializer="glorot_normal",kernel_regularizer=regularizers.l2(0.001) )(x)

x = BatchNormalization()(x)

x=Dropout(0.5)(x)

x = Dense(75,activation="relu", kernel_initializer="he_normal",kernel_regularizer=regularizers.l2(0.001))(x)



output = Dense(2, activation='softmax', name='output')(x)

model_3= Model(inputs=[essay, input_wot],outputs=[output])



print(model_3.summary())
```

```
WARNING:tensorflow:Layer lstm_6 will not use cuDNN kernel since it does
n't meet the cuDNN kernel criteria. It will use generic GPU kernel as fa
llback when running on GPU
Model: "functional_13"
_________________________________________________________________________
_________________________
Layer (type)                    Output Shape         Param #     Connect
ed to
=========================================================================
=========================
input_5 (InputLayer)            [(None, 250)]        0
_________________________________________________________________________
_________________________
input_6 (InputLayer)            [(None, 101, 1)]     0
_________________________________________________________________________
_________________________
embedding_26 (Embedding)        (None, 250, 300)     14439000    input_5
[0][0]
_________________________________________________________________________
_________________________
conv1d_4 (Conv1D)               (None, 99, 300)      1200        input_6
[0][0]
_________________________________________________________________________
_________________________
lstm_6 (LSTM)                   (None, 250, 200)     400800      embeddi
ng_26[0][0]
_________________________________________________________________________
_________________________
conv1d_5 (Conv1D)               (None, 97, 150)      135150      conv1d_
4[0][0]
_________________________________________________________________________
_________________________
flatten_28 (Flatten)            (None, 50000)        0           lstm_6
[0][0]
_________________________________________________________________________
_________________________
flatten_29 (Flatten)            (None, 14550)        0           conv1d_
5[0][0]
_________________________________________________________________________
_________________________
concatenate_6 (Concatenate)     (None, 64550)        0           flatten
_28[0][0]
                                                                 flatten
_29[0][0]
_________________________________________________________________________
_________________________
dense_22 (Dense)                (None, 120)          7746120     concate
nate_6[0][0]
_________________________________________________________________________
_________________________
dropout_12 (Dropout)            (None, 120)          0           dense_2
2[0][0]
_________________________________________________________________________
_________________________
dense_23 (Dense)                (None, 200)          24200       dropout
_12[0][0]
_________________________________________________________________________
_________________________
batch_normalization_6 (BatchNor (None, 200)          800         dense_2
3[0][0]
_________________________________________________________________________
```

```
__________________________________________________________________________________
dropout_13 (Dropout)            (None, 200)          0           batch_n
ormalization_6[0][0]

__________________________________________________________________________________
dense_24 (Dense)                (None, 75)           15075       dropout
_13[0][0]

__________________________________________________________________________________
output (Dense)                  (None, 2)            152         dense_2
4[0][0]
==================================================================================
==========================
Total params: 22,762,497
Trainable params: 8,323,097
Non-trainable params: 14,439,400

__________________________________________________________________________________
None
```

In [88]:

```
train = [X_train_pad,rest_train]
test = [X_test_pad,rest_test]
```

In [89]:

```
#Input layer
import warnings
warnings.filterwarnings("ignore")
import datetime
import os
checkpoint1 = ModelCheckpoint("model_3.h5",

                             monitor="val_loss",
                             mode="auto",
                             save_best_only = True,
                             verbose=1)
earlystop1 = EarlyStopping(monitor = 'val_loss',
                            mode="auto",
                            min_delta = 0,
                            patience = 4,
                            verbose = 2)

log_dir = os.path.join("logs", datetime.datetime.now().strftime("%Y%m%d-%H%M%S"))
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1,
write_graph=True,write_grads=True)
metric=Metrics(test, y_test)


callbacks_2= [checkpoint1,earlystop1,tensorboard_callback]
```

```
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the
`TensorBoard` Callback.
```

In [95]:

```
model_3.compile(optimizer='Nadam', loss='categorical_crossentropy', metrics=[auroc])
history2= model_3.fit(train, y_train, batch_size=600, epochs=20, verbose=2,callbacks=callbacks_2, validation_data=(test, y_test))
```

```
Epoch 1/20
WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow compared
to the batch time (batch time: 1.3865s vs `on_train_batch_end` time: 3.861
8s). Check your callbacks.

Epoch 00001: val_loss did not improve from 0.42303
122/122 - 130s - loss: 0.7631 - auroc: 0.6179 - val_loss: 0.7736 - val_aur
oc: 0.6781
Epoch 2/20

Epoch 00002: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.5909 - auroc: 0.7054 - val_loss: 0.6728 - val_aur
oc: 0.7203
Epoch 3/20

Epoch 00003: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.5236 - auroc: 0.7301 - val_loss: 0.6177 - val_aur
oc: 0.7386
Epoch 4/20

Epoch 00004: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.4835 - auroc: 0.7497 - val_loss: 0.5959 - val_aur
oc: 0.7504
Epoch 5/20

Epoch 00005: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.4563 - auroc: 0.7615 - val_loss: 0.5511 - val_aur
oc: 0.7518
Epoch 6/20

Epoch 00006: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.4412 - auroc: 0.7697 - val_loss: 0.5459 - val_aur
oc: 0.7506
Epoch 7/20

Epoch 00007: val_loss did not improve from 0.42303
122/122 - 126s - loss: 0.4291 - auroc: 0.7849 - val_loss: 0.4818 - val_aur
oc: 0.7580
Epoch 8/20

Epoch 00008: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.4210 - auroc: 0.7955 - val_loss: 0.5625 - val_aur
oc: 0.7539
Epoch 9/20

Epoch 00009: val_loss did not improve from 0.42303
122/122 - 124s - loss: 0.4157 - auroc: 0.8121 - val_loss: 0.5212 - val_aur
oc: 0.7536
Epoch 10/20

Epoch 00010: val_loss did not improve from 0.42303
122/122 - 123s - loss: 0.4724 - auroc: 0.5543 - val_loss: 0.4656 - val_aur
oc: 0.6632
Epoch 11/20

Epoch 00011: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.4316 - auroc: 0.6194 - val_loss: 0.5496 - val_aur
oc: 0.7419
Epoch 12/20

Epoch 00012: val_loss did not improve from 0.42303
```

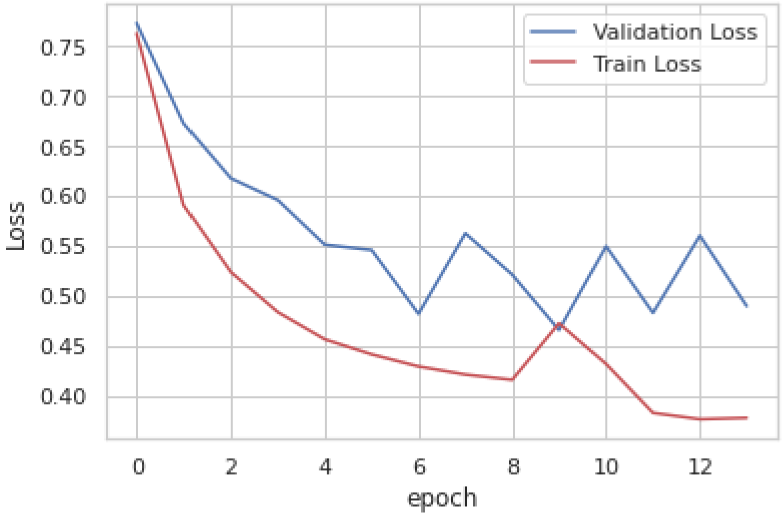
```
122/122 - 125s - loss: 0.3825 - auroc: 0.7955 - val_loss: 0.4828 - val_aur
oc: 0.7475
Epoch 13/20

Epoch 00013: val_loss did not improve from 0.42303
122/122 - 124s - loss: 0.3762 - auroc: 0.8304 - val_loss: 0.5603 - val_aur
oc: 0.7483
Epoch 14/20

Epoch 00014: val_loss did not improve from 0.42303
122/122 - 125s - loss: 0.3775 - auroc: 0.8518 - val_loss: 0.4896 - val_aur
oc: 0.7338
Epoch 00014: early stopping
```

In [97]:

```
%matplotlib inline
fig,ax = plt.subplots(1,1)
ax.set_xlabel('epoch')
ax.set_ylabel('Loss')
x = list(range(0,14))
vy = history2.history['val_loss']
ty = history2.history['loss']
ax.plot(x, vy, 'b', label="Validation Loss")
ax.plot(x, ty, 'r', label="Train Loss")
plt.legend()
```

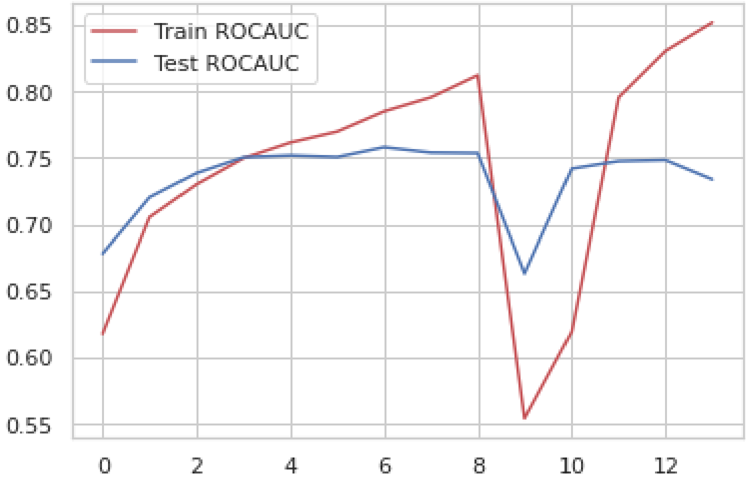
Out[97]:

```
<matplotlib.legend.Legend at 0x7f0716f7a630>
```



In [99]:

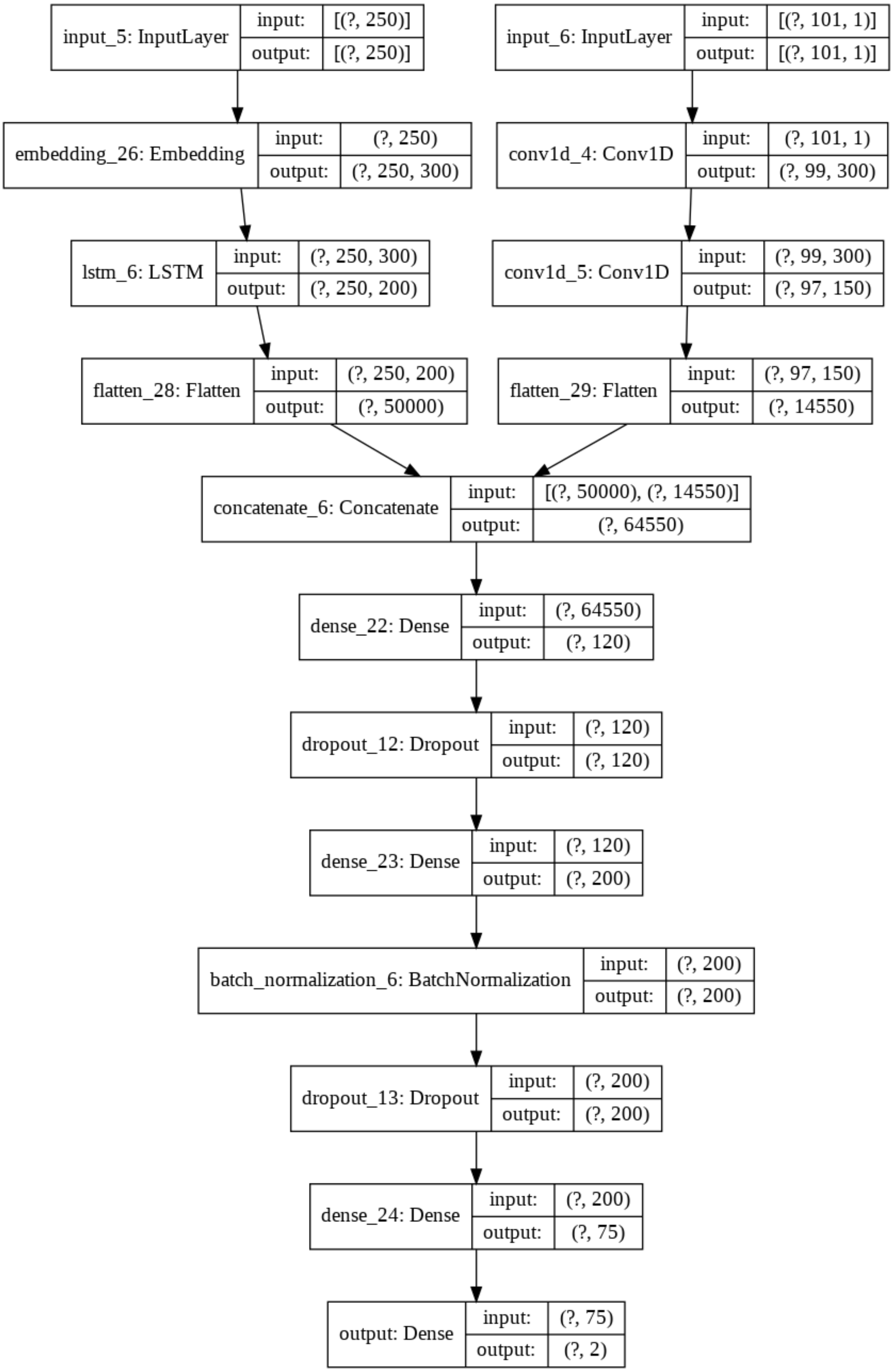
```
plt.plot(history2.history['auroc'], 'r')
plt.plot(history2.history['val_auroc'], 'b')
plt.legend({'Train ROCAUC': 'r', 'Test ROCAUC':'b'})
plt.show()
```



In [100]:

```
dot_img_file = '/tmp/model_3.png'
tf.keras.utils.plot_model(model_3, to_file=dot_img_file, show_shapes=True)
```

Out[100]:


input_5: InputLayer
input: [(?, 250)]
output: [(?, 250)]
input_6: InputLayer
input: [(?, 101, 1)]
output: [(?, 101, 1)]
embedding_26: Embedding
input: (?, 250)
output: (?, 250, 300)
conv1d_4: Conv1D
input: (?, 101, 1)
output: (?, 99, 300)
lstm_6: LSTM
input: (?, 250, 300)
output: (?, 250, 200)
conv1d_5: Conv1D
input: (?, 99, 300)
output: (?, 97, 150)
flatten_28: Flatten
input: (?, 250, 200)
output: (?, 50000)
flatten_29: Flatten
input: (?, 97, 150)
output: (?, 14550)
concatenate_6: Concatenate
input: [(?, 50000), (?, 14550)]
output: (?, 64550)
dense_22: Dense
input: (?, 64550)
output: (?, 120)
dropout_12: Dropout
input: (?, 120)
output: (?, 120)
dense_23: Dense
input: (?, 120)
output: (?, 200)
batch_normalization_6: BatchNormalization
input: (?, 200)
output: (?, 200)
dropout_13: Dropout
input: (?, 200)
output: (?, 200)
dense_24: Dense
input: (?, 200)
output: (?, 75)
output: Dense
input: (?, 75)
output: (?, 2)

In [ ]:

```
```