LSTM - Assignment

August 22, 2019

0.1 Assignment : 14

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_numer
---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 [ ]: # 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)
```

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.

0.1.5 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.

1 Assignment:

```
In [1]: %matplotlib inline
        import warnings
        warnings.filterwarnings("ignore")

        import pandas as pd
        import numpy as np
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.feature_extraction.text import CountVectorizer
        import matplotlib.pyplot as plt
        from scipy.sparse import hstack

        import re
        # Tutorial about Python regular expressions: https://pymotw.com/2/re/

        from nltk.corpus import stopwords
        import pickle

        from tqdm import tqdm
        import os
```

1.1 Reading data

```
In [2]: project_data = pd.read_csv('new_train.csv', nrows=5000)
        resource_data = pd.read_csv('resources.csv')
```

```
In [3]: print("Number of data points in train data", project_data.shape)
        print('-'*50)
        print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (5000, 21)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category' 'project_title'
 'project_essay_1' 'project_essay_2' 'project_essay_3' 'project_essay_4'
 'project_resource_summary' 'teacher_number_of_previously_posted_projects'
 'project_is_approved' 'clean_categories' 'clean_subcategories' 'essay'
 'price' 'quantity' 'is_digit_present']
```

```
In [4]: project_data.head(2)
```

```
Out[4]:    Unnamed: 0       id                        teacher_id teacher_prefix  \
        0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
        1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.

          school_state project_submitted_datetime project_grade_category  \
        0           IN        2016-12-05 13:43:57          Grades PreK-2
        1           FL        2016-10-25 09:22:10             Grades 6-8

                                                  project_title  \
        0  Educational Support for English Learners at Home
        1             Wanted: Projector for Hungry Learners

                                                project_essay_1  \
        0  My students are English learners that are work...
        1  Our students arrive to our school eager to lea...

                                                project_essay_2        ...         \
        0  \"The limits of your language are the limits o...        ...
        1  The projector we need for our school is very c...        ...

          project_essay_4                            project_resource_summary  \
        0             NaN  My students need opportunities to practice beg...
        1             NaN  My students need a projector to help with view...

          teacher_number_of_previously_posted_projects  project_is_approved  \
        0                                            0                    0
        1                                            7                    1

                       clean_categories           clean_subcategories  \
        0             Literacy_Language                  ESL Literacy
        1  History_Civics Health_Sports  Civics_Government TeamSports
```

```
                                                essay  price  quantity  \
0  My students are English learners that are work...  154.6        23
1  Our students arrive to our school eager to lea...  299.0         1

   is_digit_present
0                 0
1                 0

[2 rows x 21 columns]
```

```
In [5]: print("Number of data points in resource data", resource_data.shape)
        print(resource_data.columns.values)
        resource_data.head(2)
```

```
Number of data points in resource data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

```
Out[5]:        id                                        description  quantity  \
        0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
        1  p069063        Bouncy Bands for Desks (Blue support pipes)         3

            price
        0  149.00
        1   14.95
```

```
In [6]: x = project_data.drop(['project_is_approved','project_essay_1','project_essay_2','proj
        y = project_data['project_is_approved'].values
```

```
In [7]: x.shape
```

```
Out[7]: (5000, 16)
```

```
In [8]: y.shape
```

```
Out[8]: (5000,)
```

1.2 Splitting Data

```
In [9]: from sklearn.model_selection import train_test_split
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=
        x_train, x_cv, y_train, y_cv = train_test_split(x_train, y_train, test_size=0.2, rando
```

```
In [10]: print(x_train.shape)
         print(x_cv.shape)
         print(x_test.shape)
```

```
(3200, 16)
(800, 16)
(1000, 16)
```

1.3 Model 1:

```
In [11]: from keras.layers import Input, Embedding, LSTM, Dense, Flatten, Dropout
         from keras.models import Model
         from keras.layers.merge import concatenate
         from keras.preprocessing.text import one_hot
         from keras.preprocessing.sequence import pad_sequences
         from keras.preprocessing.text import Tokenizer
         from sklearn.preprocessing import LabelEncoder
         from numpy import asarray, zeros
         from keras.callbacks import EarlyStopping
         from sklearn.preprocessing import OneHotEncoder
         from keras.layers import Conv1D
```

```
Using TensorFlow backend.
```

1.4 Tokenizer

for essay

```
In [12]: x_train.head(2)
```

```
Out[12]:       Unnamed: 0       id                        teacher_id teacher_prefix  \
         3444      175252  p248801  400f8750770107cdbb9fbcf852a72ea            Ms.
         2063      136561  p141074  551e854422dccca4adc92781d8bd4e4a           Mrs.

              school_state project_submitted_datetime project_grade_category  \
         3444           AZ        2016-10-13 16:19:56          Grades PreK-2
         2063           PA        2016-08-03 08:47:25             Grades 3-5

                          project_title  \
         3444  Painting With a Purpose
         2063       Big Brain Benefits

                                        project_resource_summary  \
         3444  My students need an art easel and some dot mar...
         2063  My students need 21 chess sets to have the too...

               teacher_number_of_previously_posted_projects  \
         3444                                             3
         2063                                             1

                             clean_categories      clean_subcategories  \
         3444         SpecialNeeds Music_Arts  SpecialNeeds VisualArts
         2063  Literacy_Language Math_Science     Literacy Mathematics

                                                           essay   price  quantity  \
         3444  Our students come from a variety of different ...  188.24         2
```

```
2063  Of all the students in our public schools it m...    9.31        21

      is_digit_present
3444                 0
2063                 1
```

```
In [161]: t = Tokenizer()
          t.fit_on_texts(x_train['essay'])
          vocab_size   = len(t.word_index) + 1
          encoded_train_essay = t.texts_to_sequences(x_train['essay'])
```

```
In [162]: len(encoded_train_essay)
```

```
Out[162]: 3200
```

```
In [163]: encoded_cv_essay   = t.texts_to_sequences(x_cv['essay'])
          encoded_test_essay = t.texts_to_sequences(x_test['essay'])
```

```
In [164]: len(encoded_test_essay)
```

```
Out[164]: 1000
```

1.4.1 Encoding categorical features

```
In [17]: x_train.isnull().any()
```

```
Out[17]: Unnamed: 0                                      False
         id                                              False
         teacher_id                                      False
         teacher_prefix                                  False
         school_state                                    False
         project_submitted_datetime                      False
         project_grade_category                          False
         project_title                                   False
         project_resource_summary                        False
         teacher_number_of_previously_posted_projects    False
         clean_categories                                False
         clean_subcategories                             False
         essay                                           False
         price                                           False
         quantity                                        False
         is_digit_present                                False
         dtype: bool
```

Here we confirmed before encoding that there are no null values in any column of our dataframe so now we can do the encoding.

```
In [18]: # encoding categorical features using label encoder ------> https://towardsdatascienc

         le = LabelEncoder()
```

Encoding School State

```
In [19]: x_train['school_state'] = le.fit_transform(x_train['school_state'])
         print("Number of Distinct classes in school_state: ",len(le.classes_))
         x_cv['school_state']    = le.fit_transform(x_cv['school_state'])
         #print(le.classes_)
         x_test['school_state']  = le.fit_transform(x_test['school_state'])
         #print(le.classes_)
```

```
Number of Distinct classes in school_state:  51
```

Encoding Grage category

```
In [20]: x_train['project_grade_category'] = le.fit_transform(x_train['project_grade_category'
         print("Number of Distinct classes in project_grade_categories: ",len(le.classes_))
         x_cv['project_grade_category']    = le.fit_transform(x_cv['project_grade_category'])
         x_test['project_grade_category']  = le.fit_transform(x_test['project_grade_category']
```

```
Number of Distinct classes in project_grade_categories:  4
```

Encoding Clean category

```
In [21]: x_train['clean_categories'] = le.fit_transform(x_train['clean_categories'])
         print("Number of Distinct classes in clean_categories: ",len(le.classes_))
         x_cv['clean_categories']    = le.fit_transform(x_cv['clean_categories'])
         x_test['clean_categories']  = le.fit_transform(x_test['clean_categories'])
```

```
Number of Distinct classes in clean_categories:  45
```

Encoding Clean sub-category

```
In [22]: x_train['clean_subcategories'] = le.fit_transform(x_train['clean_subcategories'])
         print("Number of Distinct classes in clean_subcategories: ",len(le.classes_))
         x_cv['clean_subcategories']    = le.fit_transform(x_cv['clean_subcategories'])
         x_test['clean_subcategories']  = le.fit_transform(x_test['clean_subcategories'])
```

```
Number of Distinct classes in clean_subcategories:  219
```

Encoding Teacher prefix

```
In [23]: x_train['teacher_prefix'] = le.fit_transform(x_train['teacher_prefix'])
         print("Number of Distinct classes in teacher_prefix: ",len(le.classes_))
         x_cv['teacher_prefix']    = le.fit_transform(x_cv['teacher_prefix'])
         x_test['teacher_prefix']  = le.fit_transform(x_test['teacher_prefix'])
```

```
Number of Distinct classes in teacher_prefix:  4
```

Encoding Remaining columns

```
In [24]: remaining_cols = ['teacher_number_of_previously_posted_projects','is_digit_present']
```

```
In [25]: ohe = OneHotEncoder(sparse=False)
         ohe.fit(x_train[remaining_cols])
```

```
Out[25]: OneHotEncoder(categorical_features=None, categories=None, drop=None,
                       dtype=<class 'numpy.float64'>, handle_unknown='error',
                       n_values=None, sparse=False)
```

```
In [26]: x_train_remaining_cols = ohe.transform(x_train[remaining_cols])
         x_cv_remaining_cols    = ohe.transform(x_cv[remaining_cols])
         x_test_remaining_cols  = ohe.transform(x_test[remaining_cols])
         print(x_train_remaining_cols.shape)
         print(x_cv_remaining_cols.shape)
         print(x_test_remaining_cols.shape)
```

```
(3200, 144)
(800, 144)
(1000, 144)
```

1.5 Padding

Essay

```
In [165]: max_length   = 400
          padded_train_essay = pad_sequences(encoded_train_essay, maxlen= max_length, padding=
          print(padded_train_essay)
```

```
[[  18    4   53 ...    0    0    0]
 [   6   37    3 ...    0    0    0]
 [2244  955  987 ...    0    0    0]
 ...
 [   9    4   10 ...    0    0    0]
 [   7 6567    9 ...    0    0    0]
 [  18   21   13 ...    0    0    0]]
```

```
In [166]: max_length   = 400
          padded_cv_essay = pad_sequences(encoded_cv_essay, maxlen= max_length, padding='post'
          print(padded_cv_essay)
```

```
[[ 9  4 10 ...  0  0  0]
 [15 93 82 ...  0  0  0]
 [ 9 21 16 ...  0  0  0]
 ...
 [15 79  5 ...  0  0  0]
 [ 9  4 19 ...  0  0  0]
 [ 9  4 10 ...  0  0  0]]
```

```
In [167]: max_length   = 400
          padded_test_essay = pad_sequences(encoded_test_essay, maxlen= max_length, padding='p
          print(padded_test_essay)
```

```
[[9546 9205  330 ...    0    0    0]
 [   9    4   10 ...    0    0    0]
 [   9    4   10 ...    0    0    0]
 ...
 [ 228   16   95 ...    0    0    0]
 [ 367  278   14 ...    0    0    0]
 [   9    4   10 ...   12   26   46]]
```

1.6 Load entire glove embedding

```
In [30]: embedding_index = dict()
         f = open('glove.6B.50d.txt', encoding='utf8')
         for line in f:
             values = line.split()
             word   = values[0]
             coefs  = asarray(values[1:], dtype='float32')
             embedding_index[word] = coefs
         f.close()
         print('Loaded %s word vectors' % len(embedding_index))
```

```
Loaded 400000 word vectors
```

```
In [31]: embedding_matrix = zeros((vocab_size, 50))
         for word, i in t.word_index.items() :
             embedding_vector = embedding_index.get(word)
             if embedding_vector is not None:
                 embedding_matrix[i] = embedding_vector
```

Here we are going to use Functional API for creating our architecture.

```
In [172]: ip_seq_total_text = Input(shape = (400,), name = 'ip_total_text')
          x = Embedding(output_dim=128, input_dim=400000, input_length=400)(ip_seq_total_text)
          x = LSTM(32, return_sequences=True)(x)
          x    = Flatten()(x)

          ip_school_state = Input(shape=(1,), name= 'ip_school_state')
          y   = Embedding(output_dim=1, input_dim=51, input_length=1)(ip_school_state)
          y   = Flatten()(y)

          ip_grade_category = Input(shape=(1,), name = 'ip_grade_category')
          z = Embedding(output_dim=1, input_dim=4, input_length=1)(ip_grade_category)
          z = Flatten()(z)
```

```
ip_clean_category = Input(shape=(1,), name = 'ip_clean_category')
a = Embedding(output_dim=1, input_dim=45, input_length=1)(ip_clean_category)
a = Flatten()(a)

ip_clean_subcategory= Input(shape=(1,), name ='ip_clean_subcategory')
b = Embedding(output_dim=1, input_dim=219, input_length=1)(ip_clean_subcategory)
b = Flatten()(b)

ip_teacher_prefix= Input(shape=(1,), name ='ip_teacher_prefix')
c = Embedding(output_dim=1, input_dim=4, input_length=1)(ip_teacher_prefix)
c = Flatten()(c)

ip_combined_columns = Input(shape=(144,), name= 'ip_combined_columns')
d = Dense(64, activation='relu')(ip_combined_columns)
#"""
model = concatenate([x,y,z,a,b,c,d])
model = Dense(64, activation='relu')(model)
model = Dropout(0.5)(model)

model = Dense(32, activation='relu')(model)
model = Dropout(0.25)(model)

model = Dense(16, activation='relu')(model)
output_layer = Dense(2, activation="sigmoid")(model)

model = Model(inputs = [ip_seq_total_text,ip_school_state,ip_grade_category,ip_clean_
# Compiling the model
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
print(model.summary())
```

```
_______________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
===============================================================================================
ip_total_text (InputLayer)      (None, 400)          0
_______________________________________________________________________________________________
embedding_43 (Embedding)        (None, 400, 128)     51200000    ip_total_text[0][0]
_______________________________________________________________________________________________
ip_school_state (InputLayer)    (None, 1)            0
_______________________________________________________________________________________________
ip_grade_category (InputLayer)  (None, 1)            0
_______________________________________________________________________________________________
ip_clean_category (InputLayer)  (None, 1)            0
_______________________________________________________________________________________________
ip_clean_subcategory (InputLaye (None, 1)            0
_______________________________________________________________________________________________
ip_teacher_prefix (InputLayer)  (None, 1)            0
```

```
______________________________________________________________________________________________
lstm_8 (LSTM)                   (None, 400, 32)      20608       embedding_43[0][0]
______________________________________________________________________________________________
embedding_44 (Embedding)        (None, 1, 1)         51          ip_school_state[0][0]
______________________________________________________________________________________________
embedding_45 (Embedding)        (None, 1, 1)         4           ip_grade_category[0][0]
______________________________________________________________________________________________
embedding_46 (Embedding)        (None, 1, 1)         45          ip_clean_category[0][0]
______________________________________________________________________________________________
embedding_47 (Embedding)        (None, 1, 1)         219         ip_clean_subcategory[0][0]
______________________________________________________________________________________________
embedding_48 (Embedding)        (None, 1, 1)         4           ip_teacher_prefix[0][0]
______________________________________________________________________________________________
ip_combined_columns (InputLayer (None, 144)          0
______________________________________________________________________________________________
flatten_43 (Flatten)            (None, 12800)        0           lstm_8[0][0]
______________________________________________________________________________________________
flatten_44 (Flatten)            (None, 1)            0           embedding_44[0][0]
______________________________________________________________________________________________
flatten_45 (Flatten)            (None, 1)            0           embedding_45[0][0]
______________________________________________________________________________________________
flatten_46 (Flatten)            (None, 1)            0           embedding_46[0][0]
______________________________________________________________________________________________
flatten_47 (Flatten)            (None, 1)            0           embedding_47[0][0]
______________________________________________________________________________________________
flatten_48 (Flatten)            (None, 1)            0           embedding_48[0][0]
______________________________________________________________________________________________
dense_36 (Dense)                (None, 64)           9280        ip_combined_columns[0][0]
______________________________________________________________________________________________
concatenate_8 (Concatenate)     (None, 12869)        0           flatten_43[0][0]
                                                                 flatten_44[0][0]
                                                                 flatten_45[0][0]
                                                                 flatten_46[0][0]
                                                                 flatten_47[0][0]
                                                                 flatten_48[0][0]
                                                                 dense_36[0][0]
______________________________________________________________________________________________
dense_37 (Dense)                (None, 64)           823680      concatenate_8[0][0]
______________________________________________________________________________________________
dropout_15 (Dropout)            (None, 64)           0           dense_37[0][0]
______________________________________________________________________________________________
dense_38 (Dense)                (None, 32)           2080        dropout_15[0][0]
______________________________________________________________________________________________
dropout_16 (Dropout)            (None, 32)           0           dense_38[0][0]
______________________________________________________________________________________________
dense_39 (Dense)                (None, 16)           528         dropout_16[0][0]
______________________________________________________________________________________________
dense_40 (Dense)                (None, 2)            34          dense_39[0][0]
```

```
==============================================================================================
Total params: 52,056,533
Trainable params: 52,056,533
Non-trainable params: 0
----------------------------------------------------------------------------------------------
None
```

```
In [173]: metrics = Metrics()
          plot = model.fit([padded_train_essay, x_train['school_state'], x_train['project_grade
```

```
Train on 3200 samples, validate on 800 samples
Epoch 1/10
 - 64s - loss: 0.5215 - acc: 0.8253 - val_loss: 0.4300 - val_acc: 0.8575
Epoch 2/10
 - 34s - loss: 0.4452 - acc: 0.8472 - val_loss: 0.4080 - val_acc: 0.8575
Epoch 3/10
 - 36s - loss: 0.3810 - acc: 0.8478 - val_loss: 0.4104 - val_acc: 0.8575
Epoch 4/10
 - 36s - loss: 0.2440 - acc: 0.8847 - val_loss: 0.6198 - val_acc: 0.7925
Epoch 5/10
 - 36s - loss: 0.1130 - acc: 0.9616 - val_loss: 0.9207 - val_acc: 0.8263
Epoch 6/10
 - 38s - loss: 0.0386 - acc: 0.9888 - val_loss: 1.3011 - val_acc: 0.7987
Epoch 7/10
 - 38s - loss: 0.0288 - acc: 0.9906 - val_loss: 1.4876 - val_acc: 0.8550
Epoch 8/10
 - 38s - loss: 0.0209 - acc: 0.9944 - val_loss: 1.3930 - val_acc: 0.8050
Epoch 9/10
 - 37s - loss: 0.0050 - acc: 0.9988 - val_loss: 1.9442 - val_acc: 0.8025
Epoch 10/10
 - 37s - loss: 0.0070 - acc: 0.9994 - val_loss: 1.7189 - val_acc: 0.8237
```

```
In [143]: """
          from keras.utils import plot_model
          plot_model(model, to_file='model.png')
          """
```

2 Model 2:

Tfidf Vectorization

```
In [103]: x_train.shape
```

```
Out[103]: (3200, 16)
```

```
In [82]: tf_idf = TfidfVectorizer( use_idf=True) # min_df=10
         tf_idf.fit(x_train['essay'])
```

```
        print("some sample features",tf_idf.get_feature_names()[0:50])
        print('='*50)

        x_train_tfidf = tf_idf.transform(x_train['essay'])
        x_cv_tfidf    = tf_idf.transform(x_cv['essay'])
        x_test_tfidf  = tf_idf.transform(x_test['essay'])

        print("After featurization\n")
        print(x_train_tfidf.shape, y_train.shape)
        print(x_cv_tfidf.shape,    y_cv.shape)
        print(x_test_tfidf.shape,  y_test.shape)
```

```
some sample features ['00', '000', '01', '02', '047', '05', '0nly', '10', '100', '1000', '100m
==================================================
After featurization

(3200, 15277) (3200,)
(800, 15277) (800,)
(1000, 15277) (1000,)
```

```
In [83]: # we are converting a dictionary with word as a key, and the idf as a value
         dictionary_train = dict(zip(tf_idf.get_feature_names(), list(tf_idf.idf_)))
```

```
In [84]: len(dictionary_train)
```

```
Out[84]: 15277
```

```
In [85]: filtered_dict = dict()
         for (key, value) in dictionary_train.items():
             # Check if key is even then add pair to new dictionary
             if value < 3:
                 continue
             elif value > 6:
                 continue
             else :
                 filtered_dict[key] = value
                 # newDict[key] = value
         print('Filtered Dictionary : ')
         print(filtered_dict)
```

```
Filtered Dictionary :
{'000': 5.703922709983389, '10': 4.466048353981772, '100': 3.664046768509744, '11': 5.51587047
```

```
In [86]: len(filtered_dict)
```

```
Out[86]: 2136
```

```
In [89]: # putting these words into a set

         words = []

         for (key, value) in filtered_dict.items():
             words.append(key)
```

```
In [90]: len(words)
```

```
Out[90]: 2136
```

```
In [129]: tf_idf.fit(words)

          x_train_tfidf = tf_idf.transform(x_train['essay'])
          x_cv_tfidf    = tf_idf.transform(x_cv['essay'])
          x_test_tfidf  = tf_idf.transform(x_test['essay'])

          print("After featurization\n")
          print(x_train_tfidf.shape, y_train.shape)
          print(x_cv_tfidf.shape,    y_cv.shape)
          print(x_test_tfidf.shape,  y_test.shape)
```

```
After featurization

(3200, 2136) (3200,)
(800, 2136) (800,)
(1000, 2136) (1000,)
```

2.1 Tokenizing

```
In [130]: t = Tokenizer()
          t.fit_on_texts(words)
          vocab_size   = len(t.word_index) + 1
          encoded_train_essay = t.texts_to_sequences(x_train['essay'])
```

```
In [131]: len(encoded_train_essay)
```

```
Out[131]: 3200
```

```
In [132]: encoded_cv_essay   = t.texts_to_sequences(x_cv['essay'])
          encoded_test_essay = t.texts_to_sequences(x_test['essay'])
```

```
In [92]: """
         encoded_cv_essay   = t.texts_to_sequences(words)
         encoded_test_essay = t.texts_to_sequences(words)
         """
```

```
In [133]: len(encoded_test_essay)
```

```
Out[133]: 1000
```

2.2 Padding

```
In [147]: max_length   = 118
          padded_train_essay = pad_sequences(encoded_train_essay, maxlen= max_length, padding=
          print(padded_train_essay)
```

```
[[ 473 1201  562 ...    0    0    0]
 [1549 1688 1199 ...    0    0    0]
 [1816  950 1591 ...    0    0    0]
 ...
 [1903 1163  923 ...    0    0    0]
 [2003 2005  812 ...    0    0    0]
 [1682 2093  341 ...    0    0    0]]
```

```
In [146]: max_length   = 118
          padded_cv_essay = pad_sequences(encoded_cv_essay, maxlen= max_length, padding='post')
          print(padded_cv_essay)
```

```
[[2005 1278  319 ...    0    0    0]
 [2005  342  855 ...    0    0    0]
 [1580 1688 2110 ...    0    0    0]
 ...
 [1722 1731  637 ...    0    0    0]
 [1490   44  331 ...    0    0    0]
 [1953  676  636 ...    0    0    0]]
```

```
In [145]: max_length   = 118
          padded_test_essay = pad_sequences(encoded_test_essay, maxlen= max_length, padding='po
          print(padded_test_essay)
```

```
[[ 631 1236 1807 ...    0    0    0]
 [ 536  284  611 ...    0    0    0]
 [1253  823 1036 ...    0    0    0]
 ...
 [1662 1012  200 ...    0    0    0]
 [1762 1855  734 ...    0    0    0]
 [ 258 1013  288 ... 1375   65 1407]]
```

```
In [149]: ip_seq_total_text = Input(shape = (118,), name = 'ip_total_text')
          x = Embedding(output_dim=128, input_dim=400000, input_length=118)(ip_seq_total_text)
          x = LSTM(32, return_sequences=True)(x)
          x    = Flatten()(x)

          ip_school_state = Input(shape=(1,), name= 'ip_school_state')
          y  = Embedding(output_dim=1, input_dim=51, input_length=1)(ip_school_state)
          y  = Flatten()(y)
```

```
ip_grade_category = Input(shape=(1,), name = 'ip_grade_category')
z = Embedding(output_dim=1, input_dim=4, input_length=1)(ip_grade_category)
z = Flatten()(z)

ip_clean_category = Input(shape=(1,), name = 'ip_clean_category')
a = Embedding(output_dim=1, input_dim=45, input_length=1)(ip_clean_category)
a = Flatten()(a)

ip_clean_subcategory= Input(shape=(1,), name ='ip_clean_subcategory')
b = Embedding(output_dim=1, input_dim=219, input_length=1)(ip_clean_subcategory)
b = Flatten()(b)

ip_teacher_prefix= Input(shape=(1,), name ='ip_teacher_prefix')
c = Embedding(output_dim=1, input_dim=4, input_length=1)(ip_teacher_prefix)
c = Flatten()(c)

ip_combined_columns = Input(shape=(144,), name= 'ip_combined_columns')
d = Dense(64, activation='relu')(ip_combined_columns)
#"""
model = concatenate([x,y,z,a,b,c,d])
model = Dense(64, activation='relu')(model)
model = Dropout(0.5)(model)

model = Dense(32, activation='relu')(model)
model = Dropout(0.25)(model)

model = Dense(16, activation='relu')(model)
output_layer = Dense(2, activation="sigmoid")(model)

model = Model(inputs = [ip_seq_total_text,ip_school_state,ip_grade_category,ip_clean_
# Compiling the model
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
print(model.summary())
```

```
_______________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
===============================================================================================
ip_total_text (InputLayer)      (None, 118)          0
_______________________________________________________________________________________________
embedding_25 (Embedding)        (None, 118, 128)     51200000    ip_total_text[0][0]
_______________________________________________________________________________________________
ip_school_state (InputLayer)    (None, 1)            0
_______________________________________________________________________________________________
ip_grade_category (InputLayer)  (None, 1)            0
_______________________________________________________________________________________________
```

```
ip_clean_category (InputLayer)  (None, 1)            0
----------------------------------------------------------------------------------------------
ip_clean_subcategory (InputLaye (None, 1)            0
----------------------------------------------------------------------------------------------
ip_teacher_prefix (InputLayer)  (None, 1)            0
----------------------------------------------------------------------------------------------
lstm_5 (LSTM)                   (None, 118, 32)      20608       embedding_25[0][0]
----------------------------------------------------------------------------------------------
embedding_26 (Embedding)        (None, 1, 1)         51          ip_school_state[0][0]
----------------------------------------------------------------------------------------------
embedding_27 (Embedding)        (None, 1, 1)         4           ip_grade_category[0][0]
----------------------------------------------------------------------------------------------
embedding_28 (Embedding)        (None, 1, 1)         45          ip_clean_category[0][0]
----------------------------------------------------------------------------------------------
embedding_29 (Embedding)        (None, 1, 1)         219         ip_clean_subcategory[0][0]
----------------------------------------------------------------------------------------------
embedding_30 (Embedding)        (None, 1, 1)         4           ip_teacher_prefix[0][0]
----------------------------------------------------------------------------------------------
ip_combined_columns (InputLayer (None, 144)          0
----------------------------------------------------------------------------------------------
flatten_25 (Flatten)            (None, 3776)         0           lstm_5[0][0]
----------------------------------------------------------------------------------------------
flatten_26 (Flatten)            (None, 1)            0           embedding_26[0][0]
----------------------------------------------------------------------------------------------
flatten_27 (Flatten)            (None, 1)            0           embedding_27[0][0]
----------------------------------------------------------------------------------------------
flatten_28 (Flatten)            (None, 1)            0           embedding_28[0][0]
----------------------------------------------------------------------------------------------
flatten_29 (Flatten)            (None, 1)            0           embedding_29[0][0]
----------------------------------------------------------------------------------------------
flatten_30 (Flatten)            (None, 1)            0           embedding_30[0][0]
----------------------------------------------------------------------------------------------
dense_21 (Dense)                (None, 64)           9280        ip_combined_columns[0][0]
----------------------------------------------------------------------------------------------
concatenate_5 (Concatenate)     (None, 3845)         0           flatten_25[0][0]
                                                                 flatten_26[0][0]
                                                                 flatten_27[0][0]
                                                                 flatten_28[0][0]
                                                                 flatten_29[0][0]
                                                                 flatten_30[0][0]
                                                                 dense_21[0][0]
----------------------------------------------------------------------------------------------
dense_22 (Dense)                (None, 64)           246144      concatenate_5[0][0]
----------------------------------------------------------------------------------------------
dropout_9 (Dropout)             (None, 64)           0           dense_22[0][0]
----------------------------------------------------------------------------------------------
dense_23 (Dense)                (None, 32)           2080        dropout_9[0][0]
----------------------------------------------------------------------------------------------
```

```
dropout_10 (Dropout)            (None, 32)          0           dense_23[0][0]
____________________________________________________________________________________________________
dense_24 (Dense)                (None, 16)          528         dropout_10[0][0]
____________________________________________________________________________________________________
dense_25 (Dense)                (None, 2)           34          dense_24[0][0]
====================================================================================================
Total params: 51,478,997
Trainable params: 51,478,997
Non-trainable params: 0
____________________________________________________________________________________________________
None
```

```
In [156]: plot = model.fit([padded_train_essay, x_train['school_state'], x_train['project_grad
```

```
Train on 3200 samples, validate on 800 samples
Epoch 1/10
 - 24s - loss: 5.1854e-05 - acc: 1.0000 - val_loss: 2.8190 - val_acc: 0.7863
Epoch 2/10
 - 24s - loss: 3.0572e-04 - acc: 0.9997 - val_loss: 2.8325 - val_acc: 0.7863
Epoch 3/10
 - 23s - loss: 3.4679e-05 - acc: 1.0000 - val_loss: 2.8438 - val_acc: 0.7863
Epoch 4/10
 - 24s - loss: 4.2036e-05 - acc: 1.0000 - val_loss: 2.8557 - val_acc: 0.7850
Epoch 5/10
 - 26s - loss: 1.6915e-04 - acc: 1.0000 - val_loss: 2.8927 - val_acc: 0.7825
Epoch 6/10
 - 30s - loss: 5.5480e-05 - acc: 1.0000 - val_loss: 2.9049 - val_acc: 0.7837
Epoch 7/10
 - 27s - loss: 5.9717e-05 - acc: 1.0000 - val_loss: 2.9107 - val_acc: 0.7837
Epoch 8/10
 - 28s - loss: 4.4371e-05 - acc: 1.0000 - val_loss: 2.9051 - val_acc: 0.7837
Epoch 9/10
 - 29s - loss: 9.2774e-05 - acc: 1.0000 - val_loss: 2.9049 - val_acc: 0.7863
Epoch 10/10
 - 28s - loss: 1.8611e-04 - acc: 1.0000 - val_loss: 2.9048 - val_acc: 0.7863
```

3 Model 3:

3.1 Encoding categorical features using OneHot encoder

```
In [39]: list_of_categorical_features = ['school_state', 'teacher_prefix', 'project_grade_categ
```

```
In [40]: len(list_of_categorical_features)
```

```
Out[40]: 7
```

```
In [49]: # encoding categorical features  --------->    https://towardsdatascience.com/encodin

         #ohe = CountVectorizer()
         ohe = OneHotEncoder(sparse=False)
         ohe.fit(x_train[list_of_categorical_features])

Out[49]: OneHotEncoder(categorical_features=None, categories=None, drop=None,
                       dtype=<class 'numpy.float64'>, handle_unknown='error',
                       n_values=None, sparse=False)
```

```
In [50]: x_train_ohe = ohe.transform(x_train[list_of_categorical_features])
         x_cv_ohe    = ohe.transform(x_cv[list_of_categorical_features])
         x_test_ohe  = ohe.transform(x_test[list_of_categorical_features])
         print(x_train_ohe.shape)
         print(x_cv_ohe.shape)
         print(x_test_ohe.shape)

(3200, 467)
(800, 467)
(1000, 467)
```

Reshaping the train, cv and test data

```
In [89]: x_train_ohe = x_train_ohe.reshape(-1, 467, 1)
         x_cv_ohe    = x_cv_ohe.reshape(-1, 467, 1)
         x_test_ohe  = x_test_ohe.reshape(-1, 467, 1)
```

```
In [91]: # printing new 3 dimensional shapes of our one hot encodeded data
         print(x_train_ohe.shape)
         print(x_cv_ohe.shape)
         print(x_test_ohe.shape)

(3200, 467, 1)
(800, 467, 1)
(1000, 467, 1)
```

```
In [100]: ip_seq_total_data1 = Input(shape=(400,), name='total_data')
          x1 = Embedding(input_dim=400000, output_dim=128, input_length=400)(ip_seq_total_data
          x1 = LSTM(32, return_sequences=True)(x1)
          x1 = Flatten()(x1)

          other_than_text_data = Input(shape=(467, 1), name='no_text_data')
          y1 = Conv1D(filters=32, kernel_size=4, activation='relu')(other_than_text_data)
          y1 = Conv1D(filters=16, kernel_size=8, activation='relu')(y1)
          y1 = Flatten()(y1)

          model = concatenate([x1, y1])
          model = Dense(64, activation='relu')(model)
          model = Dropout(0.5)(model)
          model = Dense(64, activation='relu')(model)
          model = Dropout(0.25)(model)
          model = Dense(32, activation='relu')(model)
          output_layer = Dense(2, activation='softmax')(model)


          model_3 = Model(inputs = [ip_seq_total_data1,other_than_text_data], outputs= output_
          # Compiling the model
          model_3.compile(loss='sparse_categorical_crossentropy',
                        optimizer='adam',
                        metrics=['accuracy'])
          print(model_3.summary())
```

```
______________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==============================================================================================
total_data (InputLayer)         (None, 400)          0

______________________________________________________________________________________________
no_text_data (InputLayer)       (None, 467, 1)       0

______________________________________________________________________________________________
embedding_32 (Embedding)        (None, 400, 128)     51200000    total_data[0][0]

______________________________________________________________________________________________
conv1d_31 (Conv1D)              (None, 464, 32)      160         no_text_data[0][0]

______________________________________________________________________________________________
lstm_22 (LSTM)                  (None, 400, 32)      20608       embedding_32[0][0]

______________________________________________________________________________________________
conv1d_32 (Conv1D)              (None, 457, 16)      4112        conv1d_31[0][0]

______________________________________________________________________________________________
flatten_43 (Flatten)            (None, 12800)        0           lstm_22[0][0]

______________________________________________________________________________________________
flatten_44 (Flatten)            (None, 7312)         0           conv1d_32[0][0]

______________________________________________________________________________________________
concatenate_14 (Concatenate)    (None, 20112)        0           flatten_43[0][0]
                                                                 flatten_44[0][0]

______________________________________________________________________________________________
dense_53 (Dense)                (None, 64)           1287232     concatenate_14[0][0]

______________________________________________________________________________________________
dropout_27 (Dropout)            (None, 64)           0           dense_53[0][0]

______________________________________________________________________________________________
dense_54 (Dense)                (None, 64)           4160        dropout_27[0][0]

______________________________________________________________________________________________
dropout_28 (Dropout)            (None, 64)           0           dense_54[0][0]

______________________________________________________________________________________________
dense_55 (Dense)                (None, 32)           2080        dropout_28[0][0]

______________________________________________________________________________________________
```

```
dense_56 (Dense)                (None, 2)            66          dense_55[0][0]
==============================================================================================
Total params: 52,518,418
Trainable params: 52,518,418
Non-trainable params: 0
______________________________________________________________________________________________
None
```

```
In [101]: model_3.fit([padded_train_essay, x_train_ohe],y_train, epochs=10, batch_size=128, ve
```

```
Train on 3200 samples, validate on 800 samples
Epoch 1/10
 - 44s - loss: 0.4947 - acc: 0.8191 - val_loss: 0.4090 - val_acc: 0.8575
Epoch 2/10
 - 38s - loss: 0.4150 - acc: 0.8478 - val_loss: 0.4074 - val_acc: 0.8575
Epoch 3/10
 - 38s - loss: 0.3014 - acc: 0.8528 - val_loss: 0.5439 - val_acc: 0.8550
Epoch 4/10
 - 38s - loss: 0.1241 - acc: 0.9528 - val_loss: 0.6736 - val_acc: 0.8263
Epoch 5/10
 - 37s - loss: 0.0458 - acc: 0.9866 - val_loss: 0.9496 - val_acc: 0.7850
Epoch 6/10
 - 37s - loss: 0.0088 - acc: 0.9988 - val_loss: 1.5104 - val_acc: 0.8313
Epoch 7/10
 - 37s - loss: 0.0038 - acc: 0.9988 - val_loss: 1.6792 - val_acc: 0.8125
Epoch 8/10
 - 37s - loss: 0.0018 - acc: 1.0000 - val_loss: 1.8458 - val_acc: 0.8413
Epoch 9/10
 - 38s - loss: 0.0081 - acc: 0.9984 - val_loss: 1.6535 - val_acc: 0.8187
Epoch 10/10
 - 38s - loss: 0.0036 - acc: 0.9988 - val_loss: 1.5266 - val_acc: 0.8037
```

```
Out[101]: <keras.callbacks.History at 0x1ea55c4d9e8>
```