Part One

This part is intended to demostrate that there can be a neural network constructed with the input data

In [1]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import random
from tensorflow import keras
from tensorflow.keras import layers
```

In [2]:
```
#  import data
data_rs = pd.read_csv('The Global Dataset 14 Apr 2020.csv',low_memory=False)
# data_rs = pd.DataFrame(data_rs).to_numpy()

#  select columns
selected_columns = data_rs[["gender","ageBroad","RecruiterRelationship", "Coun
tryOfExploitation","isSexualExploit"]]

new_df = selected_columns.copy()


temp=new_df.drop(new_df.index[new_df['isSexualExploit'] == -99]) #drop non inf
ormation layers
new_df=temp.drop(temp.index[temp['ageBroad'] == '-99']) #drop non-information
 layers


temp=new_df.drop(new_df.index[new_df['RecruiterRelationship'] == '-99']) #drop
non information layers
new_df=temp.drop(temp.index[temp['RecruiterRelationship'] == -99]) #drop non i
nformation layers

new_df_=new_df.drop(new_df.index[new_df['CountryOfExploitation'] == '-99']) #d
rop non-information layers

cleanup_nums = {"gender":     {"Female": 2, "Male": 1},
                "ageBroad": {"0--8": 1, "9--17": 2, "18--20": 3, "21--23": 4,
                                  "24--26": 5, "27--29": 6, "30--38":7, "39--4
7":8, "48+":9 },
                "CountryOfExploitation":     {"US":1,"RU":1, "ID":2,"HT":3,
                                                "JO":4,
                                                "UG":5,
                                                "KZ":6,
                                                "LB":7,
                                                "PL":8,
                                                "SA":9,
                                                "AE":10,
                                                "BD":11,
                                                "KH":12,
                                                "CN":13,
                                                "PH":14,
                                                "LY":15,
                                                "BY":16,
                                                "AF":17,
                                                "HK":18,
                                                "EG":19,
                                                "MY":20},

                "RecruiterRelationship":     {"Not Specified":4,
                                "Other":4 ,
                                "Intimate Partner":1 ,
                                "Family/Relative":2 ,
                                "Friend/Acquaintance":3 ,
                                "Friend/Acquaintance; Other":3 ,
                                "Family/Relative; Intimate Partner":3 ,
                                "Intimate Partner; Other":1 ,
```

```
                                "Friend/Acquaintance; Intimate Partner":1 ,
                                "Not Specified; Other":4 ,
                                "Family/Relative; Other":2 ,
                                "Family/Relative; Friend/Acquaintance":2 ,
                                "Family/Relative; Not Specified":2 ,
                                "Intimate Partner; Not Specified": 1,
                                "Friend/Acquaintance; Intimate Partner; Other"
:3 ,
                                "Family/Relative; Friend/Acquaintance; Other":
2 ,
                                "Friend/Acquaintance; Intimate Partner; Not Sp
ecified":3 ,
                                "Family/Relative; Intimate Partner; Other": 2,
                                "Friend/Acquaintance; Not Specified; Other":3
,
                                "Friend/Acquaintance; Not Specified":3}      }
                # gender, ages
new_df_.replace(cleanup_nums, inplace=True) #replaces

new_df_["CountryOfExploitation"].value_counts() #find the ranges of ages
```

Out[2]:
```
1     13422
2       271
3        89
4        73
5        66
6        53
7        52
8        50
9        46
10       34
11       26
12       17
13       14
14       13
15       11
16       10
17        4
18        2
19        1
20        1
Name: CountryOfExploitation, dtype: int64
```

In [3]:
```
new_df_["RecruiterRelationship"].value_counts() #find the ranges of ages
```

Out[3]:
```
4    10486
1     1551
3     1112
2     1106
Name: RecruiterRelationship, dtype: int64
```

In [4]:
```
new_df = new_df_#displaying the head values
new_df
```

Out[4]:

	gender	ageBroad	RecruiterRelationship	CountryOfExploitation	isSexualExploit
9840	2	7	4	4	0
9842	2	7	4	4	0
9843	2	7	4	4	0
9844	2	7	4	4	0
9845	2	7	4	4	0
...	...	...	...	...	...
48768	1	2	2	1	1
48769	1	2	2	1	1
48770	1	2	2	1	1
48771	1	2	2	1	1
48772	1	2	2	1	1

14255 rows × 5 columns

In [5]:
```
new_df_["isSexualExploit"].value_counts() #find the ranges of ages
```

Out[5]:
```
1    11992
0     2263
Name: isSexualExploit, dtype: int64
```

In [6]:
```
new_df.count() #display counts table
new_df['isSexualExploit'].value_counts() #display counts sex
```

Out[6]:
```
1    11992
0     2263
Name: isSexualExploit, dtype: int64
```

In [7]:
```
is_SexualExploit =  new_df['isSexualExploit']==1 #15140
is_not_SexualExploit =  new_df['isSexualExploit']==0 #7794

datab_is_sex = new_df[is_SexualExploit] # 15140
datab_is_not_sex = new_df[is_not_SexualExploit] #7794

datab_is_sex_  = datab_is_sex.sample(frac=1).reset_index(drop=True)

datab_is_sex_ = datab_is_sex_.drop(datab_is_sex_.index[:-2263 ], axis=0)
temp = datab_is_sex[:2263]
datab_is_sex_
```

Out[7]:

	gender	ageBroad	RecruiterRelationship	CountryOfExploitation	isSexualExploit
9729	2	2	4	1	1
9730	2	2	4	1	1
9731	2	2	2	1	1
9732	2	4	4	1	1
9733	2	6	4	1	1
...	...	...	...	...	...
11987	2	2	4	1	1
11988	2	4	4	1	1
11989	2	9	4	1	1
11990	2	2	4	1	1
11991	2	5	4	1	1

2263 rows × 5 columns

In [8]:
```
t_11= pd.concat([datab_is_sex_, datab_is_not_sex], ignore_index=True)
#new_df = t_11
t_11
```

Out[8]:

	gender	ageBroad	RecruiterRelationship	CountryOfExploitation	isSexualExploit
0	2	2	4	1	1
1	2	2	4	1	1
2	2	2	2	1	1
3	2	4	4	1	1
4	2	6	4	1	1
...	...	...	...	...	...
4521	1	2	4	1	0
4522	1	2	4	1	0
4523	1	2	4	1	0
4524	1	2	4	1	0
4525	1	2	4	1	0

4526 rows × 5 columns

In [9]:
```
#t_11.plot.bar(stacked=True);
```

In [10]:
```
# spliting the data to train and test sets
train=new_df.sample(frac=0.8,random_state=200) #random state is a seed value
test=new_df.drop(train.index)
```

In [11]:
```
# get train x and train y with the label alues
train_x = train.drop('isSexualExploit',axis=1).to_numpy()
train_y = train.drop(train.columns[[0,1,2,3]], axis=1).to_numpy()
```

In [12]:
```
# get train x and train y with the label alues
test_x = test.drop('isSexualExploit',axis=1).to_numpy()
test_y = test.drop(test.columns[[0, 1,2,3]], axis=1).to_numpy()
```

In [13]:
```
test_y
```

Out[13]:
```
array([[0],
       [0],
       [0],
       ...,
       [1],
       [1],
       [1]], dtype=int64)
```

In [14]:
```
test_x
```

Out[14]:
```
array([[2, 7, 4, 4],
       [2, 7, 4, 4],
       [2, 7, 4, 4],
       ...,
       [1, 2, 2, 1],
       [1, 2, 1, 1],
       [1, 2, 2, 1]], dtype=int64)
```

In [15]:
```
# Reserve 1000 samples for validation
x_val = train_x[-1000:]
y_val = train_y[-1000:]
x_train = train_x[:-1000]
y_train = train_y[:-1000]
```

In [16]:
```
inputs = keras.Input(shape=(4,), name="int") #input = 2
x = layers.Dense(32, activation="sigmoid", name="dense_1")(inputs)
x = layers.Dense(16, activation="sigmoid", name="dense_2")(x)
x = layers.Dropout(0.25, noise_shape=None, seed=None)(x)

x = layers.Dense(32, activation="relu", name="dense_3")(x)
x = layers.Dropout(0.25, noise_shape=None, seed=None)(x)

x = layers.Dense(64, activation="sigmoid", name="dense_4")(x)

outputs = layers.Dense(2, activation="softmax", name="predictions")(x)

model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
int (InputLayer)             [(None, 4)]               0         
_________________________________________________________________
dense_1 (Dense)              (None, 32)                160       
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528       
_________________________________________________________________
dropout (Dropout)            (None, 16)                0         
_________________________________________________________________
dense_3 (Dense)              (None, 32)                544       
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0         
_________________________________________________________________
dense_4 (Dense)              (None, 64)                2112      
_________________________________________________________________
predictions (Dense)          (None, 2)                 130       
=================================================================
Total params: 3,474
Trainable params: 3,474
Non-trainable params: 0
_________________________________________________________________
```

In [17]:
```
model.compile(
    optimizer=keras.optimizers.RMSprop(),
     loss=keras.losses.SparseCategoricalCrossentropy(),
     metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

In [18]:
```
# just to demostrate this net work can work
# need more work on selecting the parameters
# because now this has very large loss val

model.fit(
    x_train,
    y_train,
    batch_size=64,
    epochs=100,
    validation_data=(x_val, y_val),
)
```

```
Epoch 1/100
163/163 [==============================] - 1s 4ms/step - loss: 0.4309 - spars
e_categorical_accuracy: 0.8386 - val_loss: 0.3728 - val_sparse_categorical_ac
curacy: 0.8430
Epoch 2/100
163/163 [==============================] - 0s 2ms/step - loss: 0.3228 - spars
e_categorical_accuracy: 0.8650 - val_loss: 0.2096 - val_sparse_categorical_ac
curacy: 0.9490
Epoch 3/100
163/163 [==============================] - 0s 2ms/step - loss: 0.2191 - spars
e_categorical_accuracy: 0.9235 - val_loss: 0.1370 - val_sparse_categorical_ac
curacy: 0.9600
Epoch 4/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1637 - spars
e_categorical_accuracy: 0.9485 - val_loss: 0.1055 - val_sparse_categorical_ac
curacy: 0.9710
Epoch 5/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1391 - spars
e_categorical_accuracy: 0.9602 - val_loss: 0.0980 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 6/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1279 - spars
e_categorical_accuracy: 0.9638 - val_loss: 0.1001 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 7/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1242 - spars
e_categorical_accuracy: 0.9638 - val_loss: 0.0974 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 8/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1201 - spars
e_categorical_accuracy: 0.9645 - val_loss: 0.0957 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 9/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1177 - spars
e_categorical_accuracy: 0.9665 - val_loss: 0.0968 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 10/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1141 - spars
e_categorical_accuracy: 0.9668 - val_loss: 0.0951 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 11/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1111 - spars
e_categorical_accuracy: 0.9673 - val_loss: 0.0946 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 12/100
163/163 [==============================] - 0s 3ms/step - loss: 0.1085 - spars
e_categorical_accuracy: 0.9670 - val_loss: 0.0936 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 13/100
163/163 [==============================] - 0s 3ms/step - loss: 0.1049 - spars
e_categorical_accuracy: 0.9684 - val_loss: 0.0942 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 14/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1067 - spars
e_categorical_accuracy: 0.9677 - val_loss: 0.0920 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 15/100
```

```
163/163 [==============================] - 0s 2ms/step - loss: 0.1038 - spars
e_categorical_accuracy: 0.9689 - val_loss: 0.0958 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 16/100
163/163 [==============================] - 0s 2ms/step - loss: 0.1060 - spars
e_categorical_accuracy: 0.9679 - val_loss: 0.0912 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 17/100
163/163 [==============================] - 0s 3ms/step - loss: 0.1052 - spars
e_categorical_accuracy: 0.9678 - val_loss: 0.0907 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 18/100
163/163 [==============================] - 1s 3ms/step - loss: 0.1005 - spars
e_categorical_accuracy: 0.9693 - val_loss: 0.0924 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 19/100
163/163 [==============================] - 1s 4ms/step - loss: 0.1001 - spars
e_categorical_accuracy: 0.9691 - val_loss: 0.0890 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 20/100
163/163 [==============================] - 1s 4ms/step - loss: 0.1006 - spars
e_categorical_accuracy: 0.9687 - val_loss: 0.0891 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 21/100
163/163 [==============================] - 1s 4ms/step - loss: 0.1014 - spars
e_categorical_accuracy: 0.9682 - val_loss: 0.0952 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 22/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0965 - spars
e_categorical_accuracy: 0.9708 - val_loss: 0.0885 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 23/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0982 - spars
e_categorical_accuracy: 0.9699 - val_loss: 0.0899 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 24/100
163/163 [==============================] - 1s 4ms/step - loss: 0.1013 - spars
e_categorical_accuracy: 0.9678 - val_loss: 0.0948 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 25/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0994 - spars
e_categorical_accuracy: 0.9685 - val_loss: 0.0888 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 26/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0967 - spars
e_categorical_accuracy: 0.9694 - val_loss: 0.0876 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 27/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0963 - spars
e_categorical_accuracy: 0.9696 - val_loss: 0.0869 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 28/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0964 - spars
e_categorical_accuracy: 0.9694 - val_loss: 0.0897 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 29/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0972 - spars
```

```
e_categorical_accuracy: 0.9697 - val_loss: 0.0904 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 30/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0928 - spars
e_categorical_accuracy: 0.9713 - val_loss: 0.0897 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 31/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0946 - spars
e_categorical_accuracy: 0.9696 - val_loss: 0.0876 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 32/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0961 - spars
e_categorical_accuracy: 0.9697 - val_loss: 0.0868 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 33/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0959 - spars
e_categorical_accuracy: 0.9704 - val_loss: 0.0864 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 34/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0946 - spars
e_categorical_accuracy: 0.9694 - val_loss: 0.0897 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 35/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0958 - spars
e_categorical_accuracy: 0.9698 - val_loss: 0.0926 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 36/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0940 - spars
e_categorical_accuracy: 0.9702 - val_loss: 0.0887 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 37/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0942 - spars
e_categorical_accuracy: 0.9711 - val_loss: 0.0860 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 38/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0934 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.1037 - val_sparse_categorical_ac
curacy: 0.9680
Epoch 39/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0933 - spars
e_categorical_accuracy: 0.9706 - val_loss: 0.0873 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 40/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0941 - spars
e_categorical_accuracy: 0.9696 - val_loss: 0.0864 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 41/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0926 - spars
e_categorical_accuracy: 0.9713 - val_loss: 0.0863 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 42/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0945 - spars
e_categorical_accuracy: 0.9700 - val_loss: 0.0885 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 43/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0933 - spars
e_categorical_accuracy: 0.9707 - val_loss: 0.0866 - val_sparse_categorical_ac
```

```
curacy: 0.9740
Epoch 44/100
163/163 [==============================] - ETA: 0s - loss: 0.0934 - sparse_ca
tegorical_accuracy: 0.970 - 1s 4ms/step - loss: 0.0934 - sparse_categorical_a
ccuracy: 0.9701 - val_loss: 0.0878 - val_sparse_categorical_accuracy: 0.9740
Epoch 45/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0936 - spars
e_categorical_accuracy: 0.9707 - val_loss: 0.0881 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 46/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0928 - spars
e_categorical_accuracy: 0.9704 - val_loss: 0.0935 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 47/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0935 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.0873 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 48/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0921 - spars
e_categorical_accuracy: 0.9704 - val_loss: 0.0861 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 49/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0911 - spars
e_categorical_accuracy: 0.9712 - val_loss: 0.0873 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 50/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0936 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.0970 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 51/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0913 - spars
e_categorical_accuracy: 0.9711 - val_loss: 0.0863 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 52/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0921 - spars
e_categorical_accuracy: 0.9706 - val_loss: 0.0853 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 53/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0932 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.0855 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 54/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0909 - spars
e_categorical_accuracy: 0.9716 - val_loss: 0.0851 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 55/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0925 - spars
e_categorical_accuracy: 0.9697 - val_loss: 0.0857 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 56/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0904 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.1134 - val_sparse_categorical_ac
curacy: 0.9680
Epoch 57/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0922 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.0854 - val_sparse_categorical_ac
curacy: 0.9740
```

```
Epoch 58/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0910 - spars
e_categorical_accuracy: 0.9706 - val_loss: 0.0964 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 59/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0919 - spars
e_categorical_accuracy: 0.9711 - val_loss: 0.0913 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 60/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0920 - spars
e_categorical_accuracy: 0.9717 - val_loss: 0.0872 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 61/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0928 - spars
e_categorical_accuracy: 0.9693 - val_loss: 0.0855 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 62/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0929 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.0856 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 63/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0906 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.0880 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 64/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0919 - spars
e_categorical_accuracy: 0.9705 - val_loss: 0.0851 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 65/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0911 - spars
e_categorical_accuracy: 0.9714 - val_loss: 0.0848 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 66/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0904 - spars
e_categorical_accuracy: 0.9717 - val_loss: 0.0968 - val_sparse_categorical_ac
curacy: 0.9690
Epoch 67/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0905 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.0855 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 68/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0914 - spars
e_categorical_accuracy: 0.9717 - val_loss: 0.0898 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 69/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0920 - spars
e_categorical_accuracy: 0.9699 - val_loss: 0.0994 - val_sparse_categorical_ac
curacy: 0.9690
Epoch 70/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0906 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.0879 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 71/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0913 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.0885 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 72/100
```

```
163/163 [==============================] - 1s 4ms/step - loss: 0.0906 - spars
e_categorical_accuracy: 0.9723 - val_loss: 0.0874 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 73/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0906 - spars
e_categorical_accuracy: 0.9712 - val_loss: 0.0847 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 74/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0903 - spars
e_categorical_accuracy: 0.9712 - val_loss: 0.0871 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 75/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0906 - spars
e_categorical_accuracy: 0.9714 - val_loss: 0.0902 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 76/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0896 - spars
e_categorical_accuracy: 0.9705 - val_loss: 0.0956 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 77/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0915 - spars
e_categorical_accuracy: 0.9707 - val_loss: 0.0867 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 78/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0908 - spars
e_categorical_accuracy: 0.9719 - val_loss: 0.0845 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 79/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0899 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.0851 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 80/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0897 - spars
e_categorical_accuracy: 0.9710 - val_loss: 0.0845 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 81/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0903 - spars
e_categorical_accuracy: 0.9707 - val_loss: 0.0921 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 82/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0899 - spars
e_categorical_accuracy: 0.9713 - val_loss: 0.0880 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 83/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0894 - spars
e_categorical_accuracy: 0.9713 - val_loss: 0.0905 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 84/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0910 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.0839 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 85/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0903 - spars
e_categorical_accuracy: 0.9715 - val_loss: 0.0866 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 86/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0908 - spars
```

```
e_categorical_accuracy: 0.9702 - val_loss: 0.0859 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 87/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0914 - spars
e_categorical_accuracy: 0.9698 - val_loss: 0.0929 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 88/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0908 - spars
e_categorical_accuracy: 0.9702 - val_loss: 0.0844 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 89/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0896 - spars
e_categorical_accuracy: 0.9712 - val_loss: 0.0932 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 90/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0898 - spars
e_categorical_accuracy: 0.9710 - val_loss: 0.0950 - val_sparse_categorical_ac
curacy: 0.9690
Epoch 91/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0908 - spars
e_categorical_accuracy: 0.9709 - val_loss: 0.1046 - val_sparse_categorical_ac
curacy: 0.9690
Epoch 92/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0908 - spars
e_categorical_accuracy: 0.9712 - val_loss: 0.0876 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 93/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0886 - spars
e_categorical_accuracy: 0.9711 - val_loss: 0.0869 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 94/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0883 - spars
e_categorical_accuracy: 0.9712 - val_loss: 0.0841 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 95/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0906 - spars
e_categorical_accuracy: 0.9706 - val_loss: 0.0857 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 96/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0911 - spars
e_categorical_accuracy: 0.9708 - val_loss: 0.0866 - val_sparse_categorical_ac
curacy: 0.9730
Epoch 97/100
163/163 [==============================] - 1s 3ms/step - loss: 0.0886 - spars
e_categorical_accuracy: 0.9710 - val_loss: 0.0881 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 98/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0896 - spars
e_categorical_accuracy: 0.9699 - val_loss: 0.0878 - val_sparse_categorical_ac
curacy: 0.9700
Epoch 99/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0902 - spars
e_categorical_accuracy: 0.9697 - val_loss: 0.0850 - val_sparse_categorical_ac
curacy: 0.9740
Epoch 100/100
163/163 [==============================] - 1s 4ms/step - loss: 0.0893 - spars
```

```
e_categorical_accuracy: 0.9724 - val_loss: 0.0838 - val_sparse_categorical_ac
curacy: 0.9740
```

Out[18]:
```
<tensorflow.python.keras.callbacks.History at 0x162ccc8c208>
```

Part Two

This part is intended to demostrate the second part with text understanding

In [19]:
```
import nltk
import random
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Flatten, Embedding, LSTM, GRU
from tensorflow.keras.models import Sequential
from nltk import CFG
from nltk.corpus import nps_chat
from nltk.parse.generate import generate, demo_grammar
```

In [20]:
```
grammar = CFG.fromstring(demo_grammar)

chatroom = nps_chat.posts('10-19-20s_706posts.xml')
data = []
for p in nps_chat.xml_posts():
    data.append({"class":p.get("class"), "txt": p.text})
df = pd.DataFrame.from_dict(data)
k =df[df['class'] == 'Statement'].copy()
len_k = len(k)
k.head()
```

Out[20]:

	class	txt
0	Statement	now im left with this gay name
4	Statement	ah well
11	Statement	26/ m/ ky women that are nice please pm me
14	Statement	there ya go 10-19-20sUser7
20	Statement	i'll thunder clap your ass.

In [21]:
```
#  select columns
selected_columns = k[["txt"]]

df1 = selected_columns.copy()
m = np.zeros((len_k))

df2 =  pd.DataFrame(np.array(m),
                   columns=['label'])
df1
df2
```

Out[21]:

	label
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0
...	...
3180	0.0
3181	0.0
3182	0.0
3183	0.0
3184	0.0

3185 rows × 1 columns

In [22]:
```
def generate_sample(grammar, prod, txt):
    if prod in grammar._lhs_index:
        derivations = grammar._lhs_index[prod]
        derivation = random.choice(derivations)
        for d in derivation._rhs:
            generate_sample(grammar, d, txt)
    elif prod in grammar._rhs_index:
        # terminal
        txt.append(str(prod))
```

In [23]:
```
# generating the second half of the data with CFG
from random import choice
grammar = nltk.CFG.fromstring("""
            S -> NP VP
            PP -> P NP
            NP -> Det N | Det N PP | 'I'
            VP -> V NP | VP PP
            Det -> 'an' | 'my'|'young'
            N -> 'girl ' | 'women'| 'chick'| 'China'
            V -> 'shot' | 'want' |'service'|'sex'
            P -> 'in'
            """)

txt = []
text_array = []


df1_ = pd.DataFrame([],
                   columns=['txt'])
for j in range(len_k ):
    txt = []
    generate_sample(grammar, grammar.start(), txt)
    txt = ' '.join(word for word in txt)
    df1_ = df1_.append({'txt': txt}, ignore_index=True)
k = np.ones(len(df1_))

df2_ =  pd.DataFrame(np.array(k),
                   columns=['label'])

df1_
```

Out[23]:

	txt
0	my girl shot an girl in an China
1	an women in young girl shot I
2	an women want I in I
3	young women in I sex my girl in my girl in m...
4	young China service I
...	...
3180	I sex my women in an women
3181	my China service an girl in young girl in my...
3182	an China sex an women in an chick in I
3183	an women want I
3184	my China want an women in I in I in my China i...

3185 rows × 1 columns

In [24]:
```
## again this is the concate step with labels ones and zeros, the df texts one
and two

df1 = pd.concat([df1, df1_], axis=0)
df2 = pd.concat([df2, df2_], axis=0)

#reset the indexes for concat purpose
df1.reset_index(inplace=True, drop=True)

df2.reset_index(inplace=True, drop=True)
df2
final = pd.concat([df1, df2], axis=1)
final
```

Out[24]:

	txt	label
0	now im left with this gay name	0.0
1	ah well	0.0
2	26/ m/ ky women that are nice please pm me	0.0
3	there ya go 10-19-20sUser7	0.0
4	i'll thunder clap your ass.	0.0
...	...	...
6365	I sex my women in an women	1.0
6366	my China service an girl in young girl in my...	1.0
6367	an China sex an women in an chick in I	1.0
6368	an women want I	1.0
6369	my China want an women in I in I in my China i...	1.0

6370 rows × 2 columns

In [25]:
```
tokenizer = Tokenizer(num_words=2500, lower=True,split=' ')
tokenizer.fit_on_texts(final['txt'].values)
X = tokenizer.texts_to_sequences(final['txt'].values)
X = pad_sequences(X)
X =  pd.DataFrame(np.array(X))
Y = (final['label']).values

Y =  pd.DataFrame(np.array(Y),columns=['label'])
X
```

Out[25]:

	0	1	2	3	4	5	6	7	8	9	...	51	52	53	54	55	56	57	58	59	60
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	56	33	217	51	81	144	167
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	496	48
2	0	0	0	0	0	0	0	0	0	0	...	98	57	1016	7	28	67	89	233	36	20
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	59	82	74	15	21	497
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	152	1538	1539	52	293
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6365	0	0	0	0	0	0	0	0	0	0	...	0	0	0	2	13	3	7	1	4	7
6366	0	0	0	0	0	0	0	0	0	0	...	9	1	5	9	1	5	9	1	5	9
6367	0	0	0	0	0	0	0	0	0	0	...	4	8	13	4	7	1	4	6	1	2
6368	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	4	7	11	2
6369	0	0	0	0	0	0	0	0	0	0	...	4	7	1	4	8	1	2	1	4	8

6370 rows × 61 columns

In [26]:
```
np.shape(X)
```

Out[26]: (6370, 61)

In [27]:
```
nnn = pd.concat([X, Y], axis=1)
# np.shape(nnn)
nnn[1]
```

Out[27]:
```
0       0
1       0
2       0
3       0
4       0
       ..
6365    0
6366    0
6367    0
6368    0
6369    0
Name: 1, Length: 6370, dtype: int32
```

In [28]:
```
train=nnn.sample(frac=0.8,random_state=200) #random state is a seed value

test=nnn.drop(train.index)
train
```

Out[28]:

	0	1	2	3	4	5	6	7	8	9	...	52	53	54	55	56	57	58	59	60
2448	0	0	0	0	0	0	0	0	0	0	...	22	493	19	66	74	16	1000	63	46
4130	0	0	0	0	0	0	0	0	0	0	...	1	4	8	1	5	7	13	5	7
2766	0	0	0	0	0	0	0	0	0	0	...	0	0	150	395	18	164	14	22	112
6000	0	0	0	0	0	0	0	0	0	0	...	8	1	5	7	1	2	10	3	6
1300	0	0	0	0	0	0	0	0	0	0	...	17	450	714	19	197	60	102	1	574
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5403	0	0	0	0	0	0	0	0	0	0	...	8	1	5	9	1	3	7	1	2
1988	0	0	0	0	0	0	0	0	0	0	...	408	165	556	360	16	368	27	118	1409
3526	0	0	0	0	0	0	0	0	0	0	...	1	2	1	2	1	3	7	1	2
3994	0	0	0	0	0	0	0	0	0	0	...	9	1	5	8	1	5	9	1	2
4707	0	0	0	0	0	0	0	0	0	0	...	2	1	2	1	4	7	1	5	8

5096 rows × 62 columns

In [29]:
```
[a,v] = np.shape(train)
#v-2
train_x = train.drop('label',axis=1).to_numpy()
train_y = train.drop(train.iloc[:, 0:(v-1)], axis=1).to_numpy()
[a,v] = np.shape(test)
#v-2
test_x = test.drop('label',axis=1).to_numpy()
test_y = test.drop(test.iloc[:, 0:(v-1)], axis=1).to_numpy()

# Reserve 1000 samples for validation
x_val = train_x[-1000:]
y_val = train_y[-1000:]
x_train = train_x[:-1000]
y_train = train_y[:-1000]
```

In [30]:
```
x_val
```

Out[30]:
```
array([[0, 0, 0, ..., 1, 4, 9],
       [0, 0, 0, ..., 8, 1, 2],
       [0, 0, 0, ..., 7, 1, 2],
       ...,
       [0, 0, 0, ..., 7, 1, 2],
       [0, 0, 0, ..., 9, 1, 2],
       [0, 0, 0, ..., 1, 5, 8]])
```

In [31]:
```
y_val
```

Out[31]:
```
array([[1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
```

```
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
```

```
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
```

```
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
```

```
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
```

```
            [1.],
            [1.],
            [1.],
            [1.],
            [0.],
            [0.],
            [1.],
            [0.],
            [0.],
            [0.],
            [0.],
            [1.],
            [1.],
            [1.],
            [1.],
            [0.],
            [1.],
            [1.],
            [1.],
            [0.],
            [1.],
            [0.],
            [0.],
            [0.],
            [0.],
            [1.],
            [1.],
            [0.],
            [1.],
            [1.],
            [1.],
            [0.],
            [1.],
            [1.],
            [1.],
            [0.],
            [1.],
            [0.],
            [1.],
            [0.],
            [1.],
            [1.],
            [0.],
            [0.],
            [0.],
            [0.],
            [1.],
            [0.],
            [1.],
            [1.],
            [0.],
            [1.],
            [1.],
            [0.],
            [1.],
            [0.],
            [1.],
```

```
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
```

```
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
```

```
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
```

```
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
```

```
       [1.],
       [1.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
```

```
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
```

```
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
```

```
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
```

```
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[1.],
```

```
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
```

```
[1.],
[0.],
[0.],
[0.],
[1.],
[0.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[1.],
[0.],
[1.],
[1.],
[1.],
[0.],
[1.],
[0.],
[1.],
[1.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[0.],
[1.],
[0.],
[1.],
[1.],
```

```
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [1.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [1.],
       [1.],
       [0.],
       [1.],
       [1.],
       [1.]])
```

In [32]:
```
embed_dim =128
lstm_out = 200
print("check size of input ",v-1)
inputs = keras.Input(shape=(v-1,), name="int")
x = layers.Embedding(2500, embed_dim )(inputs)
x = layers.LSTM(lstm_out)(x)

outputs = layers.Dense(2, activation="softmax", name="predictions")(x)

model = keras.Model(inputs=inputs, outputs=outputs)
model.summary()
```

```
check size of input  61
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
int (InputLayer)             [(None, 61)]              0         
_________________________________________________________________
embedding (Embedding)        (None, 61, 128)           320000    
_________________________________________________________________
lstm (LSTM)                  (None, 200)               263200    
_________________________________________________________________
predictions (Dense)          (None, 2)                 402       
=================================================================
Total params: 583,602
Trainable params: 583,602
Non-trainable params: 0
_________________________________________________________________
```

In [33]:
```
model.compile(
    optimizer=keras.optimizers.RMSprop(),
     loss=keras.losses.SparseCategoricalCrossentropy(),
     metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

In [34]:
```
model.fit(x_train, y_train, batch_size =32,epochs=10,
    validation_data=(x_val, y_val),)
```

```
Epoch 1/10
128/128 [==============================] - 16s 129ms/step - loss: 0.0933 - sp
arse_categorical_accuracy: 0.9722 - val_loss: 0.0016 - val_sparse_categorical
_accuracy: 0.9990
Epoch 2/10
128/128 [==============================] - 15s 118ms/step - loss: 0.0194 - sp
arse_categorical_accuracy: 0.9968 - val_loss: 6.8333e-04 - val_sparse_categor
ical_accuracy: 1.0000
Epoch 3/10
128/128 [==============================] - 15s 120ms/step - loss: 0.0038 - sp
arse_categorical_accuracy: 0.9990 - val_loss: 0.0014 - val_sparse_categorical
_accuracy: 0.9990
Epoch 4/10
128/128 [==============================] - 15s 120ms/step - loss: 8.3944e-05
- sparse_categorical_accuracy: 1.0000 - val_loss: 0.0018 - val_sparse_categor
ical_accuracy: 0.9990
Epoch 5/10
128/128 [==============================] - 15s 120ms/step - loss: 0.0133 - sp
arse_categorical_accuracy: 0.9976 - val_loss: 4.8817e-04 - val_sparse_categor
ical_accuracy: 1.0000
Epoch 6/10
128/128 [==============================] - 16s 125ms/step - loss: 6.6342e-06
- sparse_categorical_accuracy: 1.0000 - val_loss: 0.0012 - val_sparse_categor
ical_accuracy: 0.9990
Epoch 7/10
128/128 [==============================] - 16s 123ms/step - loss: 0.0027 - sp
arse_categorical_accuracy: 0.9993 - val_loss: 0.0016 - val_sparse_categorical
_accuracy: 0.9990
Epoch 8/10
128/128 [==============================] - 16s 125ms/step - loss: 5.4768e-04
- sparse_categorical_accuracy: 0.9998 - val_loss: 0.0060 - val_sparse_categor
ical_accuracy: 0.9990
Epoch 9/10
128/128 [==============================] - 16s 126ms/step - loss: 1.0486e-06
- sparse_categorical_accuracy: 1.0000 - val_loss: 4.5341e-04 - val_sparse_cat
egorical_accuracy: 1.0000
Epoch 10/10
128/128 [==============================] - 16s 123ms/step - loss: 3.1461e-08
- sparse_categorical_accuracy: 1.0000 - val_loss: 0.0025 - val_sparse_categor
ical_accuracy: 0.9990
```

Out[34]:
```
<tensorflow.python.keras.callbacks.History at 0x162d964d1c8>
```

Reference:

[1] https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47 (https://towardsdatascience.com/understanding-lstm-and-its-quick-implementation-in-keras-for-sentiment-analysis-af410fd85b47)

[2] http://www.nltk.org/book_1ed/ch05.html (http://www.nltk.org/book_1ed/ch05.html)

[3] https://stackoverflow.com/questions/15009656/how-to-use-nltk-to-generate-sentences-from-an-induced-grammar (https://stackoverflow.com/questions/15009656/how-to-use-nltk-to-generate-sentences-from-an-induced-grammar)

In [ ]: