Loading data and preprocessing

In [61]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from time import time
warnings.filterwarnings("ignore")
%matplotlib inline
from sklearn.model_selection import train_test_split
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
from sklearn.metrics import f1_score, confusion_matrix
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Flatten, Embedding, Input, Dropout, LSTM, Bidirectional
from keras.utils.np_utils import to_categorical
from keras.models import load_model
from tensorflow.python.keras.callbacks import TensorBoard
from prettytable import PrettyTable as pt
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [ ]:

```
from google.colab import drive

drive.mount('/content/drive')
```

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

In [ ]:

```
%cd /content/drive/MyDrive/kaggle_toxic/
```

```
/content/drive/MyDrive/kaggle_toxic
```

In [ ]:

```
all_data = pd.read_csv('all_data.csv')
```

In [ ]:

```
all_data.head(5)
```

Out[ ]:

	id	comment_text	split	created_date	publication_id	parent_id	article_id	rating	funny	wow	sad	likes	disagree	toxicity	sev
0	1083994	He got his money... now he lies in wait till a...	train	2017-03-06 15:21:53.675241+00	21	NaN	317120	approved	0	0	0	2	0	0.373134	
1	650904	Mad dog will surely put the liberals in mental...	train	2016-12-02 16:44:21.329535+00	21	NaN	154086	approved	0	0	1	2	0	0.605263	
2	5902188	And Trump continues his lifelong cowardice by ...	train	2017-09-05 19:05:32.341360+00	55	NaN	374342	approved	1	0	2	3	7	0.666667	
3	7084460	"while arresting a man for resisting arrest".\...	test	2016-11-01 16:53:33.561631+00	13	NaN	149218	approved	0	0	0	0	0	0.815789	
4	5410943	Tucker and Paul are both total bad ass mofo's.	train	2017-06-14 05:08:21.997315+00	21	NaN	344096	approved	0	0	0	1	0	0.550000	

In [ ]:

```
toxic = []
#making comments which have probability more than 0.5 as toxic and marking them as 1 while non-toxic as 0
for i in all_data['toxicity']:
     if i > 0.5:
        toxic.append(1)
     else:
        toxic.append(0)

all_data['toxic_binary'] = toxic
```

In [ ]:

```
all_data['sub_toxic'] = all_data[['severe_toxicity','obscene','sexual_explicit', 'identity_attack','insult',
```

In [ ]:

```
sub_toxic = []
for j in range(len(all_data)):
   if all_data['toxic_binary'][j] == 1:
      if all_data['sub_toxic'][j] == 'severe_toxicity':
         sub_toxic.append(6)
      if all_data['sub_toxic'][j] == 'obscene':
         sub_toxic.append(5)
      if all_data['sub_toxic'][j] == 'sexual_explicit':
         sub_toxic.append(4)
      if all_data['sub_toxic'][j] == 'identity_attack':
         sub_toxic.append(3)
      if all_data['sub_toxic'][j] == 'insult':
         sub_toxic.append(2)
      if all_data['sub_toxic'][j] == 'threat':
         sub_toxic.append(1)
   if all_data['toxic_binary'][j] == 0:
       sub_toxic.append(0)

all_data['sub_toxic'] = sub_toxic
```

In [ ]:

```
stop = set(stopwords.words('english'))

def clean(text):
  text_token = word_tokenize(text)
  filtered_text = ' '.join([w.lower() for w in text_token if w.lower() not in stop and len(w) > 2])
  filtered_text = filtered_text.replace(r"[^a-zA-Z]+", '')
  text_only = re.sub(r'\b\d+\b', '', filtered_text)
  clean_text = text_only.replace(',', '').replace('.', '').replace(':', '')
  return clean_text
```

In [ ]:

```
all_data['clean_comment'] = [clean(str(x)) for x in all_data['comment_text']]
```

Splitting Data

In [ ]:

```
train = all_data.loc[all_data['split']=='train']
train.head(5)
```

Out[ ]:

	id	comment_text	split	created_date	publication_id	parent_id	article_id	rating	funny	wow	sad	likes	disagree	toxicity	sev
0	1083994	He got his money... now he lies in wait till a...	train	2017-03-06 15:21:53.675241+00	21	NaN	317120	approved	0	0	0	2	0	0.373134	
1	650904	Mad dog will surely put the liberals in mental...	train	2016-12-02 16:44:21.329535+00	21	NaN	154086	approved	0	0	1	2	0	0.605263	
2	5902188	And Trump continues his lifelong cowardice by ...	train	2017-09-05 19:05:32.341360+00	55	NaN	374342	approved	1	0	2	3	7	0.666667	
4	5410943	Tucker and Paul are both total bad ass mofo's.	train	2017-06-14 05:08:21.997315+00	21	NaN	344096	approved	0	0	0	1	0	0.550000	
5	6290444	Cry me a river, why don't you.\nDrinking, drug...	train	2017-11-04 22:04:11.596185+00	54	6290143.0	396946	rejected	0	0	0	0	0	0.203390	

In [ ]:

```
test = all_data.loc[all_data['split']=='test']
test.head(5)
```

Out[ ]:

	id	comment_text	split	created_date	publication_id	parent_id	article_id	rating	funny	wow	sad	likes	disagree	toxicity	se
3	7084460	"while arresting a man for resisting arrest".\...	test	2016-11-01 16:53:33.561631+00	13	NaN	149218	approved	0	0	0	0	0	0.815789	
10	7141509	NO ! There are no alternative facts. Go check...	test	2017-01-30 02:53:48.012277+00	21	919529.0	164687	approved	1	0	0	0	0	0.597222	
11	7077814	the more you whine sore loser Artster\n\nthe m...	test	2016-12-03 00:17:42.300700+00	54	649753.0	154126	approved	0	0	0	0	0	0.650000	
38	7147990	There's rarely opportunity to agree with Benne...	test	2017-09-13 16:37:16.990602+00	102	NaN	377304	approved	1	0	0	1	2	0.111111	
42	7008066	The Law has every freedom to be an asss!	test	2017-07-09 07:03:44.153492+00	54	5556167.0	353158	approved	0	0	0	0	0	0.800000	

In [ ]:

```
train = train.reset_index(drop=True)
test = test.reset_index(drop=True)
```

In [ ]:

```
X = train['clean_comment']
Y = train['sub_toxic']
```

In [ ]:

```
x_train, x_test, y_train, y_test = train_test_split(X.values,Y.values, test_size=0.2, stratify=Y)
```

In [ ]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[ ]:

```
((1443900,), (360975,), (1443900,), (360975,))
```

In [ ]:

```
train_labels = to_categorical(y_train)
val_labels = to_categorical(y_test)
```

Tokenizing and Padding

In [ ]:

```
token = Tokenizer()
token.fit_on_texts(x_train)
vocab_size = len(token.word_index) + 1
```

In [ ]:

```
train_encoded = token.texts_to_sequences(x_train)
val_encoded = token.texts_to_sequences(x_test)
```

In [ ]:

```
max_len = len(max(x_train, key = len))
max_len
```

Out[ ]:

```
1300
```

In [ ]:

```
train_padded_comment = pad_sequences(train_encoded, maxlen=max_len, padding='post')
val_padded_comment = pad_sequences(val_encoded, maxlen=max_len, padding='post')
```

In [ ]:

```
test_encoded = token.texts_to_sequences(test['clean_comment'].values)
```

In [ ]:

```
test_padded_comment = pad_sequences(test_encoded, maxlen=max_len, padding='post')
```

Simple Neural Network

In [32]:

```
model = Sequential()
model.add(Input(shape=(max_len,)))
model.add(Embedding(vocab_size,
                           128,
                           input_length = max_len,
                           name = 'embeddings'))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(7, activation="softmax"))
```

In [33]:

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embeddings (Embedding)      (None, 1300, 128)         40699904

 flatten_1 (Flatten)         (None, 166400)            0

 dense_2 (Dense)             (None, 128)               21299328

 dense_3 (Dense)             (None, 7)                 903

=================================================================
Total params: 62,000,135
Trainable params: 62,000,135
Non-trainable params: 0
_________________________________________________________________
None
```

In [35]:

```
tensorboard = TensorBoard(log_dir='logs/simv2/{}'.format(time()), histogram_freq=1, write_graph =True, update_
earlystopping = EarlyStopping(monitor='val_loss', patience=2, verbose=0, mode='min')
mcp = ModelCheckpoint('model.hdf5', save_best_only=True, monitor='val_loss', mode='min')
reducelr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2, verbose=1, mode='min')
```

In [36]:

```
history = model.fit(train_padded_comment, train_labels, epochs=5, batch_size=64, validation_data=(val_padded_c
```

```
Epoch 1/5
22561/22561 [==============================] - 500s 22ms/step - loss: 0.1509 - categorical_accuracy: 0.9573 -
val_loss: 0.1329 - val_categorical_accuracy: 0.9596 - lr: 0.0010
Epoch 2/5
22561/22561 [==============================] - 495s 22ms/step - loss: 0.1024 - categorical_accuracy: 0.9661 -
val_loss: 0.1454 - val_categorical_accuracy: 0.9578 - lr: 0.0010
Epoch 3/5
22560/22561 [============================>.] - ETA: 0s - loss: 0.0589 - categorical_accuracy: 0.9800
Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
22561/22561 [==============================] - 486s 22ms/step - loss: 0.0589 - categorical_accuracy: 0.9800 -
val_loss: 0.1938 - val_categorical_accuracy: 0.9549 - lr: 0.0010
```

In [41]:

```
%load_ext tensorboard
```

In [43]:

```
%tensorboard --logdir logs/simv2/1644935833.3039317/
```

In [37]:

```
prediction = model.predict(test_padded_comment)
```

In [38]:

```
classes =np.argmax(prediction,axis=1)
```

In [58]:

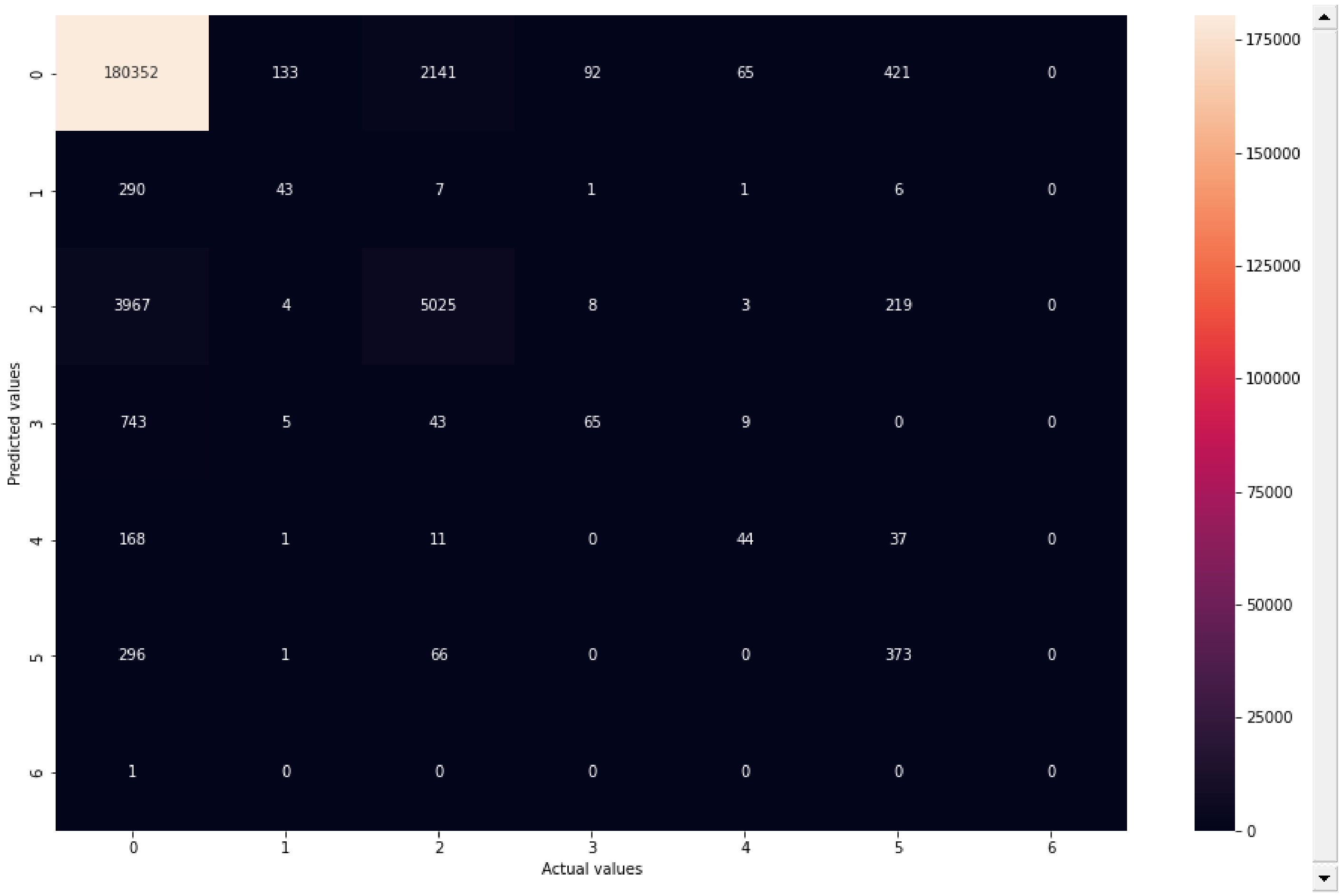
```
fone1 = f1_score(test['sub_toxic'].values, classes, average=None)
fone1
```

Out[58]:

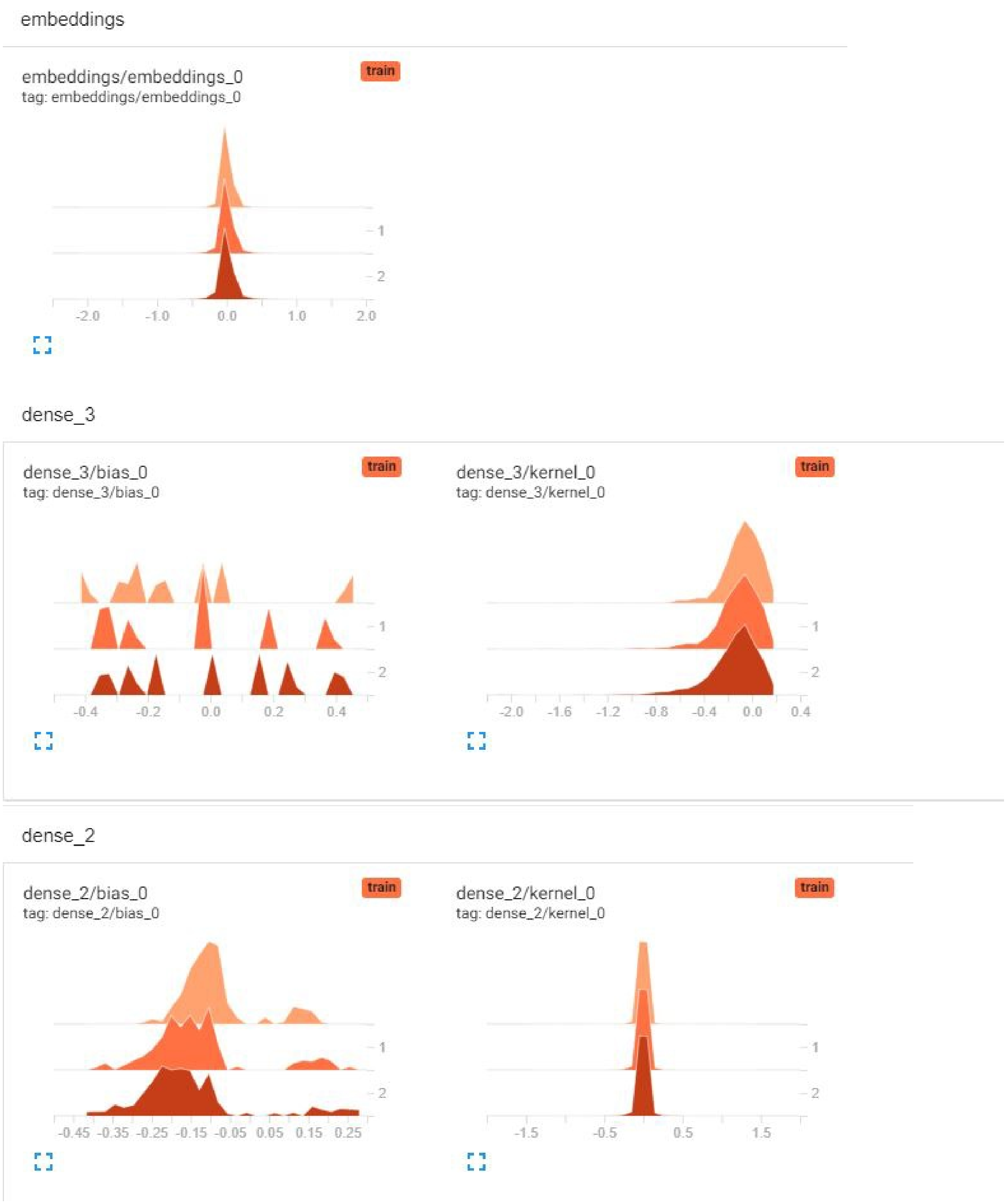
```
array([0.97746199, 0.16074766, 0.60839034, 0.12609117, 0.22976501,
       0.41629464, 0.        ])
```

In [40]:

```
confusion = confusion_matrix(test['sub_toxic'].values, classes)
plt.figure(figsize = (16,10))
sns.heatmap(confusion, annot=True, fmt='g')
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.show()
```



	0	1	2	3	4	5	6
0	180352	133	2141	92	65	421	0
1	290	43	7	1	1	6	0
2	3967	4	5025	8	3	219	0
3	743	5	43	65	9	0	0
4	168	1	11	0	44	37	0
5	296	1	66	0	0	373	0
6	1	0	0	0	0	0	0



Analysis : Model 1 </h1>

The accuracy seems inflated because of the fact that there are lots of non-toxic comments. On the other hand, f1 score on toxic categories is pretty less.

Seeing the histograms as well, we can observe that the weights are not changing much over the iterations.

LSTM with dropout

In [ ]:

```
model2 = Sequential()
model2.add(Input(shape=(max_len,)))
model2.add(Embedding(vocab_size,
                           128,
                           input_length = max_len,
                           name = 'embeddings'))
model2.add(LSTM(128, return_sequences=True,name='lstm_layer'))
model2.add(Dropout(0.1))
model2.add(LSTM(128, return_sequences=True,name='lstm_layer_2'))
model2.add(Dropout(0.1))
model2.add(Flatten())
model2.add(Dense(128, activation='relu'))
model2.add(Dense(7, activation="softmax"))
```

In [ ]:

```
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print(model2.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embeddings (Embedding)      (None, 1300, 128)         40774912  
                                                                 
 lstm_layer (LSTM)           (None, 1300, 128)         131584    
                                                                 
 dropout (Dropout)           (None, 1300, 128)         0         
                                                                 
 lstm_layer_2 (LSTM)         (None, 1300, 128)         131584    
                                                                 
 dropout_1 (Dropout)         (None, 1300, 128)         0         
                                                                 
 flatten_1 (Flatten)         (None, 166400)            0         
                                                                 
 dense_2 (Dense)             (None, 128)               21299328  
                                                                 
 dense_3 (Dense)             (None, 7)                 903       
                                                                 
=================================================================
Total params: 62,338,311
Trainable params: 62,338,311
Non-trainable params: 0
_________________________________________________________________
None
```

In [ ]:

```
tensorboard = TensorBoard(log_dir='logs/lstmv2/{}'.format(time()), histogram_freq=1, write_graph =True, update
earlystopping = EarlyStopping(monitor='val_loss', patience=2, verbose=0, mode='min')
mcp = ModelCheckpoint('model2.hdf5', save_best_only=True, monitor='val_loss', mode='min')
reducelr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2, verbose=1, mode='min')
```

In [ ]:

```
history2 = model2.fit(train_padded_comment, train_labels, epochs=8, batch_size=64, validation_data=(val_padde
```

```
Epoch 1/8
22561/22561 [==============================] - 3828s 169ms/step - loss: 0.1299 - categorical_accuracy: 0.9596
- val_loss: 0.1162 - val_categorical_accuracy: 0.9619 - lr: 0.0010
Epoch 2/8
22561/22561 [==============================] - 3837s 170ms/step - loss: 0.1042 - categorical_accuracy: 0.9640
- val_loss: 0.1147 - val_categorical_accuracy: 0.9615 - lr: 0.0010
Epoch 3/8
22561/22561 [==============================] - 3837s 170ms/step - loss: 0.0882 - categorical_accuracy: 0.9695
- val_loss: 0.1219 - val_categorical_accuracy: 0.9603 - lr: 0.0010
Epoch 4/8
22561/22561 [==============================] - ETA: 0s - loss: 0.0722 - categorical_accuracy: 0.9751
Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
22561/22561 [==============================] - 3945s 175ms/step - loss: 0.0722 - categorical_accuracy: 0.9751
- val_loss: 0.1381 - val_categorical_accuracy: 0.9550 - lr: 0.0010
```

In [56]:

```
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [57]:

```
%tensorboard --logdir logs/lstmv2/1644860717.161058/
```

In [51]:

```
model2 = load_model('model2.hdf5')
```

In [52]:

```
prediction2 = model2.predict(test_padded_comment)
```

In [53]:

```
classes2 =np.argmax(prediction2,axis=1)
```

In [59]:

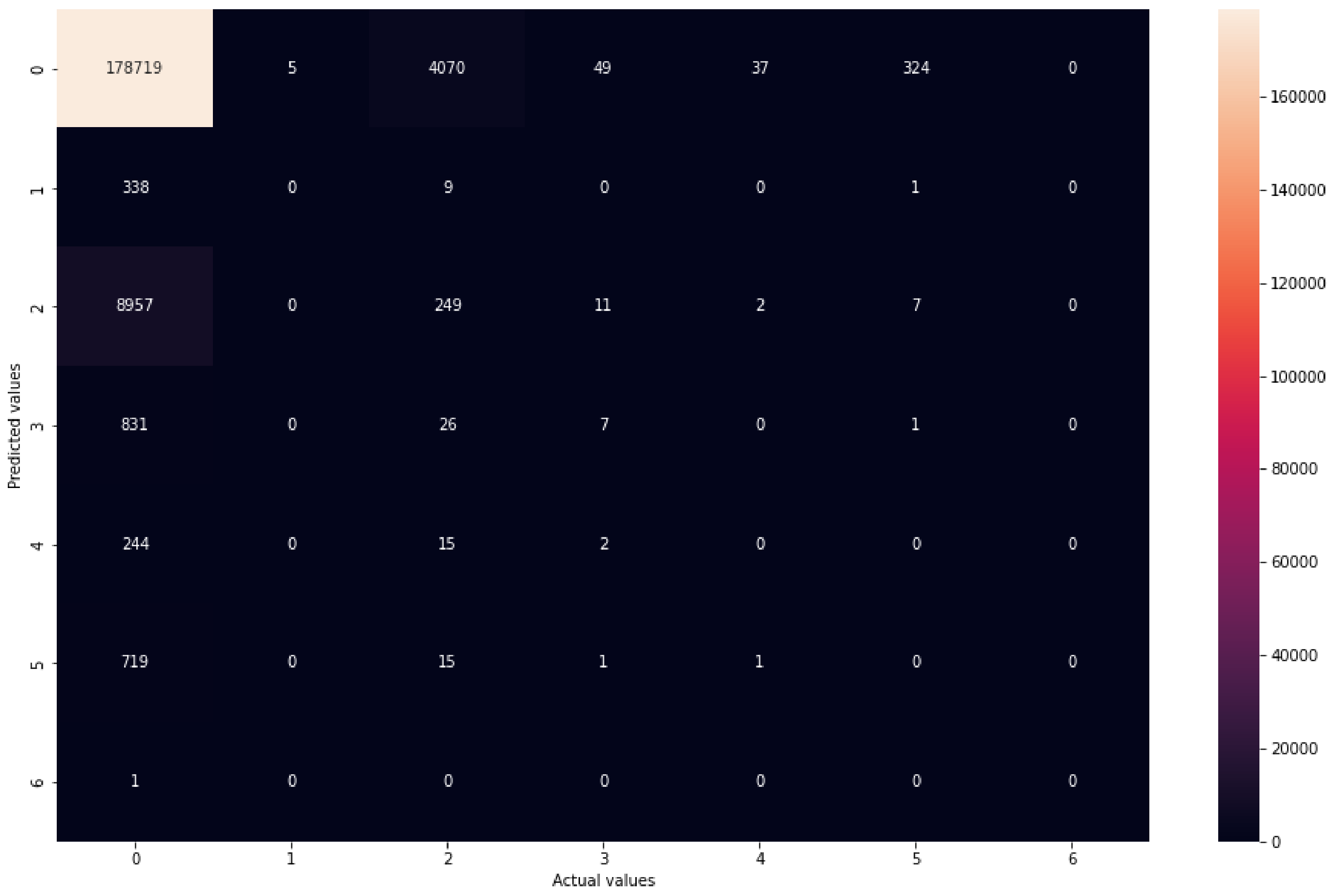
```
fone2 = f1_score(test['sub_toxic'].values, classes2, average=None)
fone2
```

Out[59]:

```
array([0.95824542, 0.        , 0.03659074, 0.01497326, 0.        ,
       0.        , 0.        ])
```

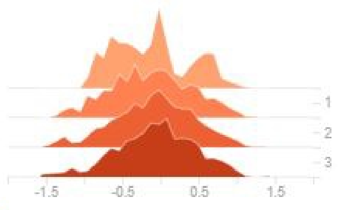
In [55]:

```
confusion = confusion_matrix(test['sub_toxic'].values, classes2)
plt.figure(figsize = (16,10))
sns.heatmap(confusion, annot=True, fmt='g')
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.show()
```

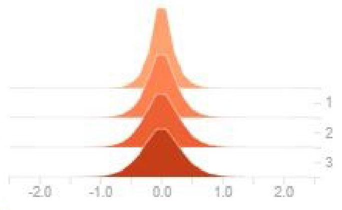


lstm_layer

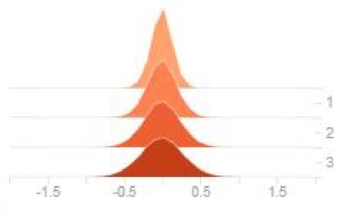
lstm_layer/lstm_cell/bias_0 train
tag: lstm_layer/lstm_cell/bias_0



lstm_layer/lstm_cell/kernel_0 train
tag: lstm_layer/lstm_cell/kernel_0

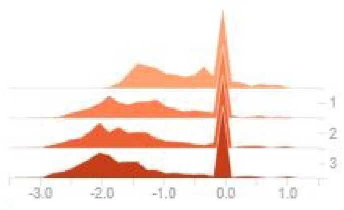


lstm_layer/lstm_cell/recurrent_kernel_0 train
tag: lstm_layer/lstm_cell/recurrent_kernel_0

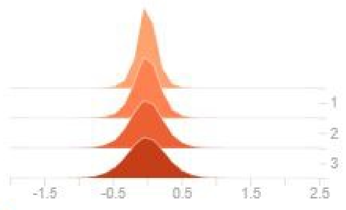


lstm_layer_2

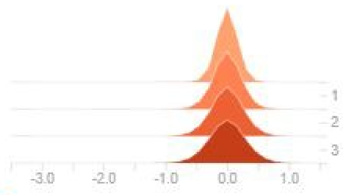
lstm_layer_2/lstm_cell_1/bias_0 train
tag: lstm_layer_2/lstm_cell_1/bias_0



lstm_layer_2/lstm_cell_1/kernel_0 train
tag: lstm_layer_2/lstm_cell_1/kernel_0

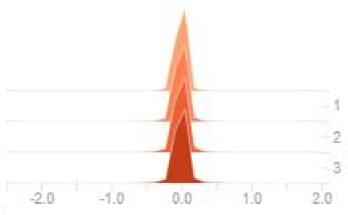


lstm_layer_2/lstm_cell_1/recurrent_kernel_0 train
tag: lstm_layer_2/lstm_cell_1/recurrent_kernel_0



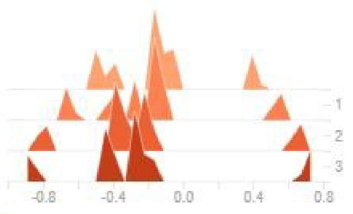
embeddings

embeddings/embeddings_0 train
tag: embeddings/embeddings_0

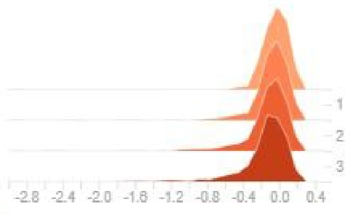


dense_3

dense_3/bias_0 train
tag: dense_3/bias_0

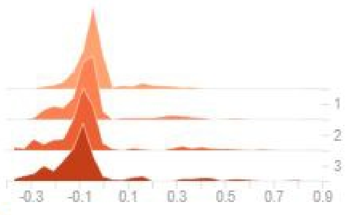


dense_3/kernel_0 train
tag: dense_3/kernel_0

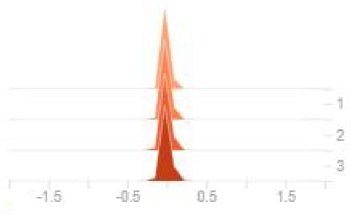


dense_2

dense_2/bias_0 train
tag: dense_2/bias_0



dense_2/kernel_0 train
tag: dense_2/kernel_0



Analysis: Model 2

This model is not performing well in classifying between the categories. The f1-score of this model is pretty less.

It can be seen in the histograms that they have smoothened out over the iterations but there is not massive difference in the weights.

Bidirectional LSTM

In [ ]:

```
model3 = Sequential()
model3.add(Input(shape=(max_len,)))
model3.add(Embedding(vocab_size,
                           128,
                           input_length = max_len,
                           name = 'embeddings'))
model3.add(Bidirectional(LSTM(128, return_sequences=True,name='bilstm_layer')))
model3.add(Dropout(0.1))
model3.add(Bidirectional(LSTM(128, return_sequences=True,name='bilstm_layer_2')))
model3.add(Dropout(0.1))
model3.add(Flatten())
model3.add(Dense(128, activation='relu'))
model3.add(Dense(7, activation="softmax"))
```

In [ ]:

```
model3.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print(model3.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embeddings (Embedding)      (None, 1300, 128)         40699904  
                                                                 
 bidirectional (Bidirectiona  (None, 1300, 256)        263168    
 l)                                                              
                                                                 
 dropout (Dropout)           (None, 1300, 256)         0         
                                                                 
 bidirectional_1 (Bidirectio  (None, 1300, 256)        394240    
 nal)                                                            
                                                                 
 dropout_1 (Dropout)         (None, 1300, 256)         0         
                                                                 
 flatten (Flatten)           (None, 332800)            0         
                                                                 
 dense (Dense)               (None, 128)               42598528  
                                                                 
 dense_1 (Dense)             (None, 7)                 903       
                                                                 
=================================================================
Total params: 83,956,743
Trainable params: 83,956,743
Non-trainable params: 0
_________________________________________________________________
None
```

In [ ]:

```
tensorboard = TensorBoard(log_dir='logs/bilstm/{}'.format(time()), histogram_freq=1, write_graph =True, update
earlystopping = EarlyStopping(monitor='val_loss', patience=2, verbose=0, mode='min')
mcp = ModelCheckpoint('model3.hdf5', save_best_only=True, monitor='val_loss', mode='min')
reducelr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2, verbose=1, mode='min')
```

In [27]:

```
history3 = model3.fit(train_padded_comment, train_labels, epochs=8, batch_size=64, validation_data=(val_padde
Epoch 1/8
22561/22561 [==============================] - 8406s 372ms/step - loss: 0.1397 - categorical_accuracy: 0.9584
- val_loss: 0.1212 - val_categorical_accuracy: 0.9604 - lr: 0.0010
Epoch 2/8
22561/22561 [==============================] - 8402s 372ms/step - loss: 0.1064 - categorical_accuracy: 0.9637
- val_loss: 0.1175 - val_categorical_accuracy: 0.9608 - lr: 0.0010
Epoch 3/8
22561/22561 [==============================] - 8437s 374ms/step - loss: 0.0880 - categorical_accuracy: 0.9698
- val_loss: 0.1212 - val_categorical_accuracy: 0.9592 - lr: 0.0010
Epoch 4/8
22561/22561 [==============================] - ETA: 0s - loss: 0.0695 - categorical_accuracy: 0.9762
Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.00010000000474974513.
22561/22561 [==============================] - 8378s 371ms/step - loss: 0.0695 - categorical_accuracy: 0.9762
- val_loss: 0.1357 - val_categorical_accuracy: 0.9563 - lr: 0.0010
```

In [48]:

```
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [49]:

```
%tensorboard --logdir logs/bilstm/1644900361.857247/
```

In [44]:

```
prediction3 = model3.predict(test_padded_comment)
```

In [45]:

```
classes3 = np.argmax(prediction3,axis=1)
```

In [60]:

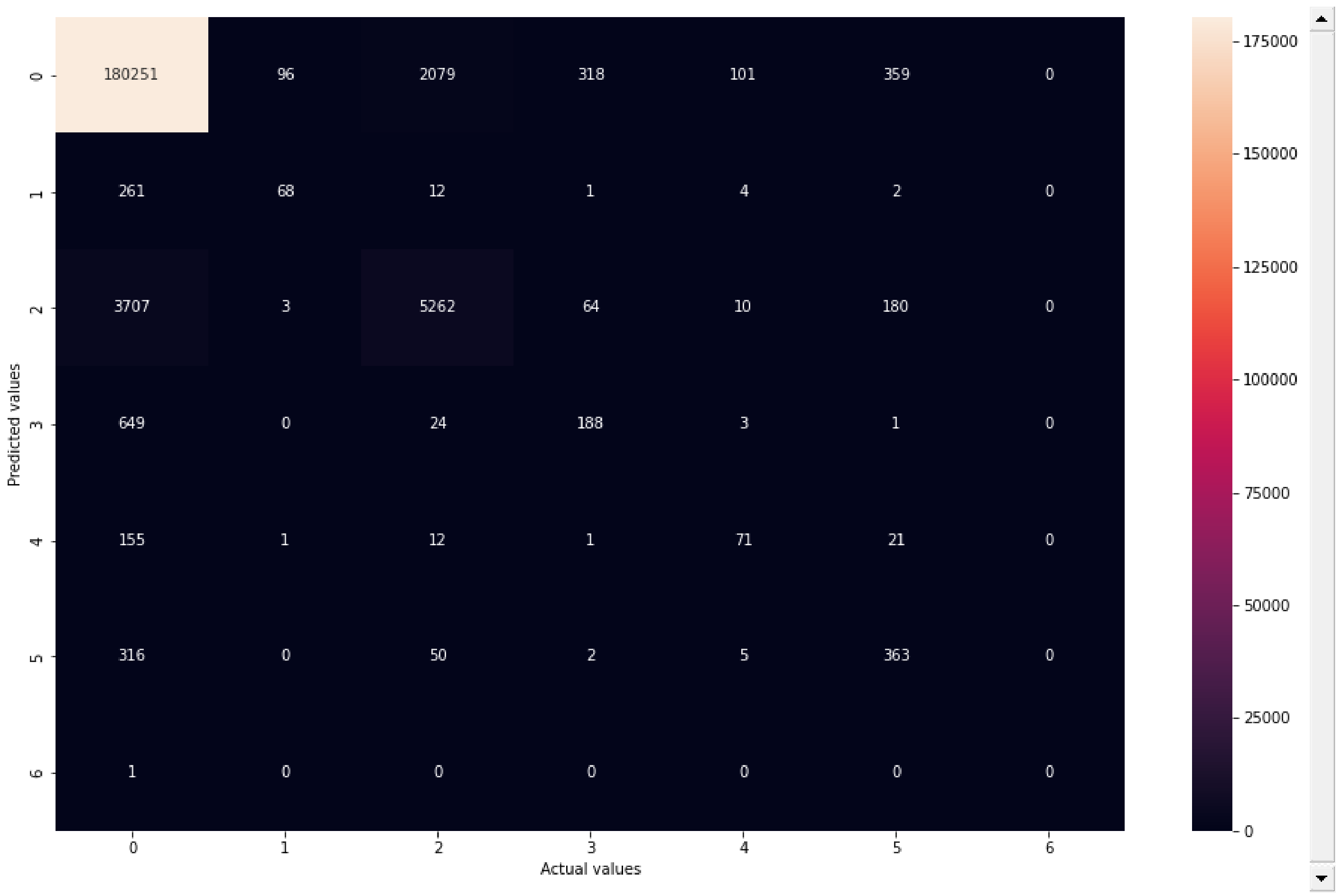
```
fone3 = f1_score(test['sub_toxic'].values, classes3, average=None)
fone3
```

Out[60]:

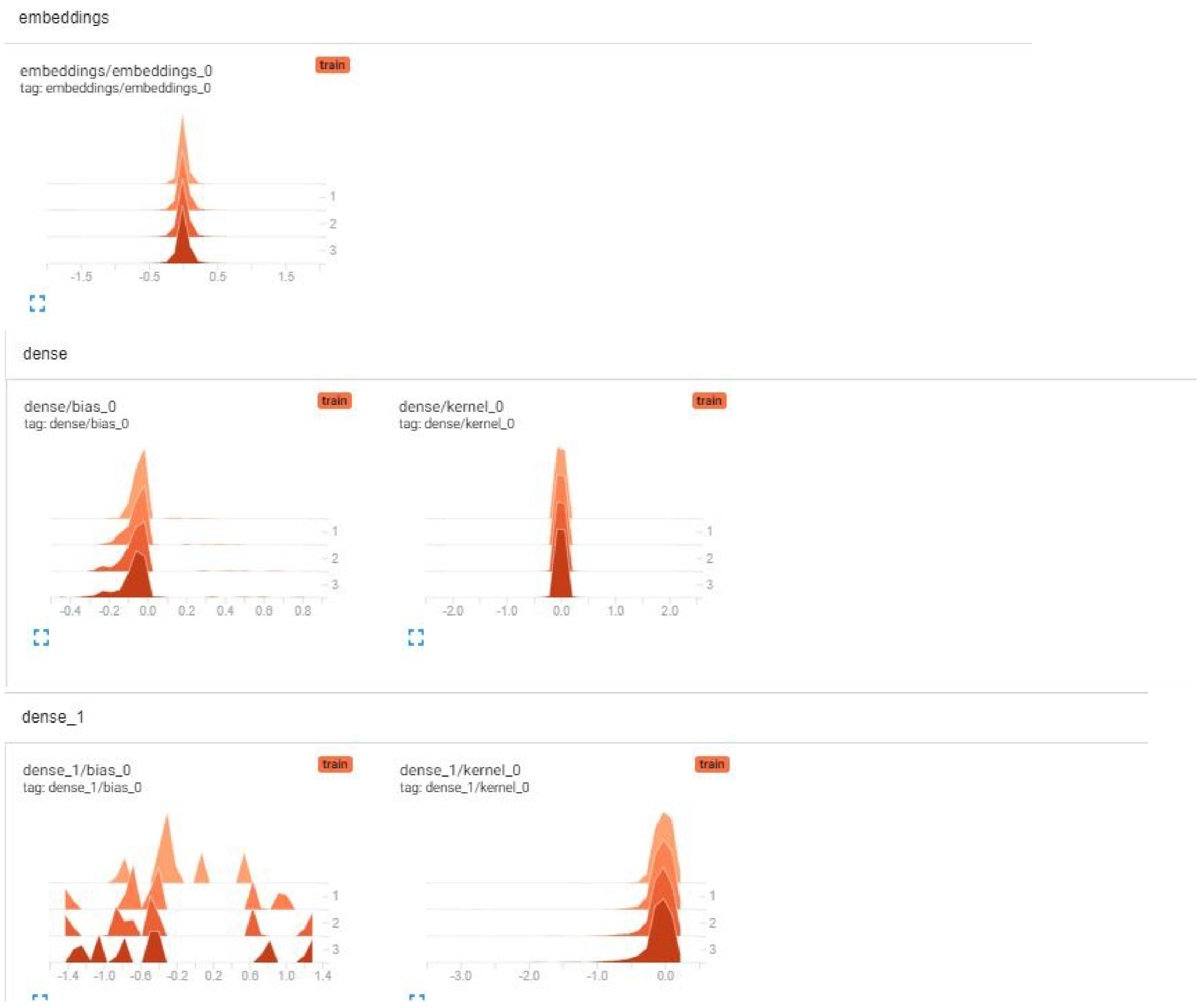
```
array([0.978179  , 0.26356589, 0.63150315, 0.26129256, 0.31208791,
       0.4368231 , 0.        ])
```

In [47]:

```
confusion = confusion_matrix(test['sub_toxic'].values, classes3)
plt.figure(figsize = (16,10))
sns.heatmap(confusion, annot=True, fmt='g')
plt.xlabel('Actual values')
plt.ylabel('Predicted values')
plt.show()
```



Predicted values \ Actual values	0	1	2	3	4	5	6
0	180251	96	2079	318	101	359	0
1	261	68	12	1	4	2	0
2	3707	3	5262	64	10	180	0
3	649	0	24	188	3	1	0
4	155	1	12	1	71	21	0
5	316	0	50	2	5	363	0
6	1	0	0	0	0	0	0



embeddings

embeddings/embeddings_0
tag: embeddings/embeddings_0

dense

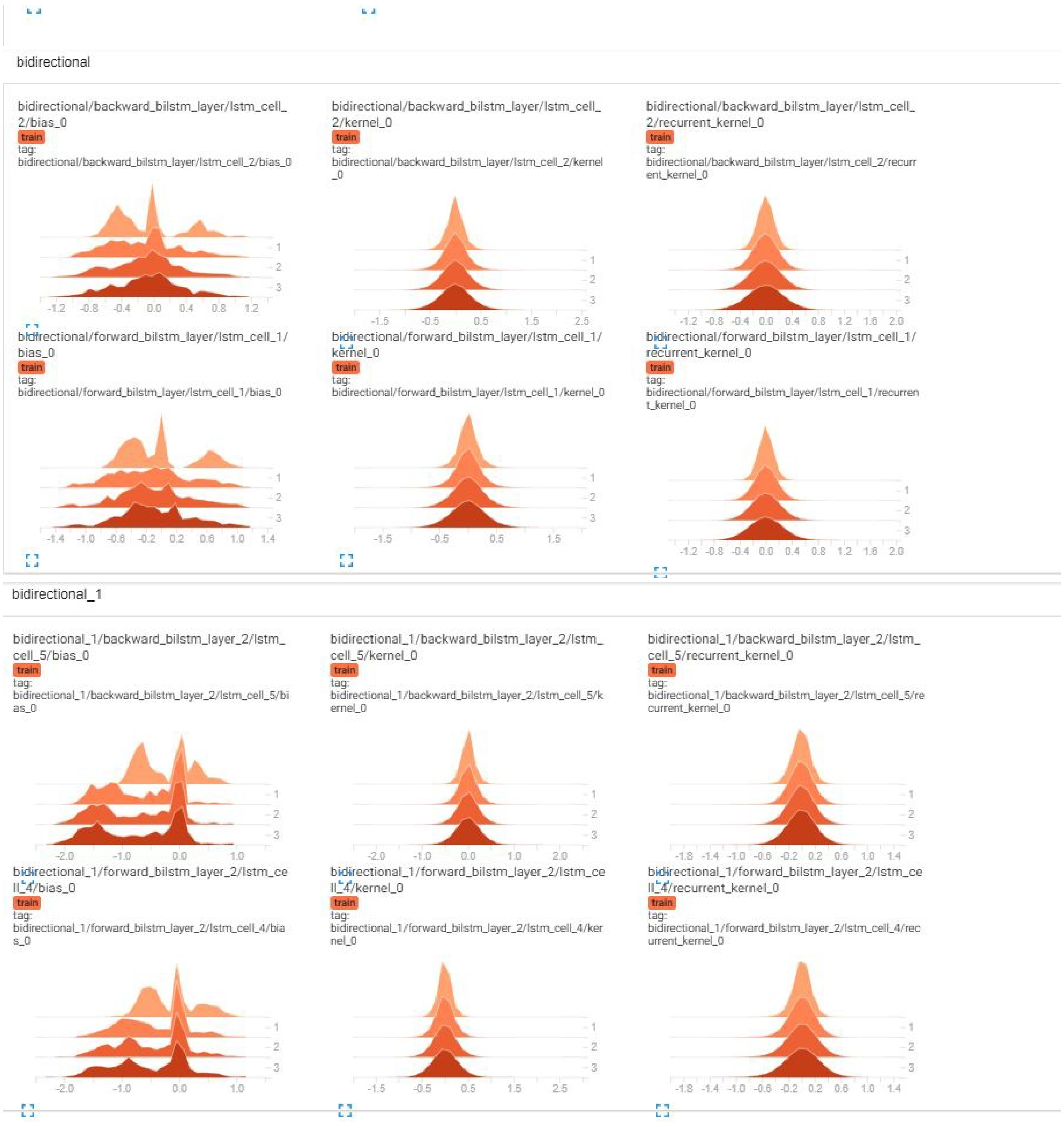
dense/bias_0
tag: dense/bias_0

dense/kernel_0
tag: dense/kernel_0

dense_1

dense_1/bias_0
tag: dense_1/bias_0

dense_1/kernel_0
tag: dense_1/kernel_0


bidirectional
bidirectional/backward_bilstm_layer/lstm_cell_2/bias_0
train
tag: bidirectional/backward_bilstm_layer/lstm_cell_2/bias_0
bidirectional/backward_bilstm_layer/lstm_cell_2/kernel_0
train
tag: bidirectional/backward_bilstm_layer/lstm_cell_2/kernel_0
bidirectional/backward_bilstm_layer/lstm_cell_2/recurrent_kernel_0
train
tag: bidirectional/backward_bilstm_layer/lstm_cell_2/recurrent_kernel_0
bidirectional/forward_bilstm_layer/lstm_cell_1/bias_0
train
tag: bidirectional/forward_bilstm_layer/lstm_cell_1/bias_0
bidirectional/forward_bilstm_layer/lstm_cell_1/kernel_0
train
tag: bidirectional/forward_bilstm_layer/lstm_cell_1/kernel_0
bidirectional/forward_bilstm_layer/lstm_cell_1/recurrent_kernel_0
train
tag: bidirectional/forward_bilstm_layer/lstm_cell_1/recurrent_kernel_0
bidirectional_1
bidirectional_1/backward_bilstm_layer_2/lstm_cell_5/bias_0
train
tag: bidirectional_1/backward_bilstm_layer_2/lstm_cell_5/bias_0
bidirectional_1/backward_bilstm_layer_2/lstm_cell_5/kernel_0
train
tag: bidirectional_1/backward_bilstm_layer_2/lstm_cell_5/kernel_0
bidirectional_1/backward_bilstm_layer_2/lstm_cell_5/recurrent_kernel_0
train
tag: bidirectional_1/backward_bilstm_layer_2/lstm_cell_5/recurrent_kernel_0
bidirectional_1/forward_bilstm_layer_2/lstm_cell_4/bias_0
train
tag: bidirectional_1/forward_bilstm_layer_2/lstm_cell_4/bias_0
bidirectional_1/forward_bilstm_layer_2/lstm_cell_4/kernel_0
train
tag: bidirectional_1/forward_bilstm_layer_2/lstm_cell_4/kernel_0
bidirectional_1/forward_bilstm_layer_2/lstm_cell_4/recurrent_kernel_0
train
tag: bidirectional_1/forward_bilstm_layer_2/lstm_cell_4/recurrent_kernel_0

Analysis : Model 3</h1>

The f1 score has improved from simple model but it is still pretty low.

Seeing the histograms as well, we can see that there is no drastic change in learning of model over the iterations.

Final results

In [62]:

```
result_table = pt(['Model Name', 'not toxic' 'threat', 'insult' 'identity_attack' 'sexual_explicit' 'obscene'
result_table = pt(['Model Name', 'not_toxic', 'threat', 'insult', 'identity_attack', 'sexual_explicit', 'obscene',
```

In [64]:

```
result_table.add_row(['NN with dropout', round(fone1[0], ndigits=3), round(fone1[1], ndigits=3), round(fone1[2
result_table.add_row(['LSTM with dense and dropout', round(fone2[0], ndigits=3), round(fone2[1], ndigits=3), r
result_table.add_row(['Bidirectional LSTM with dropout', round(fone3[0], ndigits=3), round(fone3[1], ndigits=3
```

In [65]:

```
print(result_table)
```

```
+---------------------------------+-----------+--------+--------+-----------------+-----------------+---------
+------------------+
|            Model Name           | not_toxic | threat | insult | identity_attack | sexual_explicit | obscene
| severe_toxicity |
+---------------------------------+-----------+--------+--------+-----------------+-----------------+---------
+------------------+
|         NN with dropout         |   0.977   | 0.161  | 0.608  |      0.126      |       0.23      |  0.416
|       0.0       |
|   LSTM with dense and dropout   |   0.958   |  0.0   | 0.037  |      0.015      |       0.0       |   0.0
|       0.0       |
| Bidirectional LSTM with dropout |   0.978   | 0.264  | 0.632  |      0.261      |      0.312      |  0.437
|       0.0       |
+---------------------------------+-----------+--------+--------+-----------------+-----------------+---------
+------------------+
```

Bidirectional LSTM is clearly the best model in all three of them but neural netweeok with droput is not much behind.