UML 602 Natural Language Processing

Assignment 3
(Minor Project)

Topic

TOXIC COMMENT CLASSIFIER

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Name: Kushagra Thakral
Roll No: 101703301
Group: COE14
EmailId: kushagra.thakral@gmail.com

Contact: 9034199977

Instructions to execute code

Note: I have created a Jupyter notebook for better explanation of the project and visualization of data and techniques used. It is highly recommended to view that notebook on the github link provided below.
Github link: https://github.com/Kushagra7744/Toxic-Comment-Classifier

Note: All the dependencies should be installed on your local machine.

Dependencies:
1) python3
2) numpy
3) pandas
4) plotly
5) matplotlib
6) streamlit
7) keras
8) tensorflow

Also if you want to execute the GUI. Follow the steps below:

1) clone the the above repository.
2) open terminal inside the Toxic-Comment-Classifier directory
3) type ‘streamlit run app.py’ without quotes inside terminal
a browser window will open directing to the GUI dashboard.

4) click generate random tweet checkbox
5) click process checkbox
6) click the ‘predict for random generated tweet’ button

CODE

#imports

```
import streamlit as st
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import matplotlib.pyplot as plt
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import model_from_json
from keras.layers import Dense, Input, LSTM, Embedding, Dropout,
Activation
from keras.layers import Bidirectional, GlobalMaxPool1D
from keras.models import Model
```

#display on dashboard

```
st.title('Toxic Comment Classifier')
st.sidebar.title('Toxic Comment Classifier')
st.markdown('## This dashboards visualizes the training data and predicts
toxicity of the comments')
st.markdown('### Toxicity is divided into 6 classes: toxic, severe_toxic,
obscene, threat, insult, identity_hate')
```

#loading data

```
@st.cache(persist=True)
def load_data(data_URL):
        df=pd.read_csv(data_URL)
        return df



if st.sidebar.checkbox('Load test data'):
        data_URL='data/test.csv'
        df=load_data(data_URL)
else:
```

```
    data_URL="data/train.csv"
    df=load_data(data_URL)
```

#Exploring data

```
st.sidebar.subheader('Explore data')
rows=st.sidebar.slider('Show tabular data ')
if(rows>0):
    st.markdown('**DataFrame till %d rows: **' %rows)
    st.write(df.head(rows))

if st.sidebar.button('Describe data'):
    st.markdown('**Description:**')
    st.write(df.describe())


if st.sidebar.button('Display Null values count'):
    st.markdown('**Count of NULL values for all columns:**')
    st.write(df[df.isna()==True].count())

st.sidebar.subheader('Visualize training data')
```

 # processing data to create plots

```
toxic_counts={}
for i in range(6):
   toxic_counts[df.columns[i+2]]=df[df[df.columns[i+2]]==1].count()[0]

if st.sidebar.checkbox('include normal comments'):
    normal_comments_count=(df.count())[0]-sum(toxic_counts.values())
    temp_dict={'normal':normal_comments_count}
    chart_data= {**temp_dict,**toxic_counts}
else:
    chart_data=toxic_counts
```

 # creating plots

```
select= st.sidebar.selectbox('Visualization type',['None','Pie Chart',
'Histogram'],key=2)
```

```
if select=='Pie Chart':
	fig_total=
go.Figure([go.Pie(labels=list(chart_data.keys()),values=list(chart_data.val
ues()))])
	fig_total.show()
if select=='Histogram':
	plot_val=go.Bar(x=list(chart_data.keys()),y=list(chart_data.values()))
	fig= go.Figure(plot_val)
	fig.show()

# generating random tweet
st.sidebar.subheader('Generate random tweet')
random_tweet=(df['comment_text'].sample(n=1))
if st.sidebar.checkbox('Generate random tweet',key=5):
	st.markdown('### %s'%random_tweet.iloc[0])

st.sidebar.subheader('Process tweet for prediction')

#tokenization of random tweet
def tokenize(random_tweet):
	max_features=20000
	tokenizer= Tokenizer(num_words=max_features)
	tokenizer.fit_on_texts(list(df['comment_text']))

	tokenized_comment=tokenizer.texts_to_sequences(random_tweet)
	tokenized_comment=pad_sequences(tokenized_comment,200)
	return tokenized_comment

if st.sidebar.checkbox('Process'):

	tokenized_comment=tokenize(random_tweet)
	st.write(tokenized_comment)

#loading pre-trained deep learning model

st.sidebar.subheader('Deep Learning Model')
# @st.cache(persist=True)
def load_model(m='model.json',w='weights.h5'):
	json_file=open(m,'r')
```

```
    model=json_file.read()
    json_file.close()
    model=model_from_json(model)
    model.load_weights(w)

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['acc
uracy'])
    return model

model=load_model()
```

#summary of model(will be displayed in terminal running streamlit)

```
if st.sidebar.checkbox('Model Summary'):
    st.markdown('### Check you  streamlit terminal for summary')
    st.write(model.summary())
```

#predicting toxicity for random tweet

```
if st.sidebar.button('Predict for random generated tweet'):
    st.markdown('### Here 0 indicates comment being toxic and
similarly 5 is prob of comment being identity_hate')
    st.write(model.predict(tokenized_comment))
```

Note: Model is trained using the code in jupyter-notebook. Check the github link for the same. The same model is saved in json format and retrieved in ‘app.py’.

Screenshots

1) Jupyter-notebook

Toxic Comment Classification

In [ ]:
```
#imports
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import matplotlib.pyplot as plt
```

Loading and Describing data

In [2]:
```
#loading data
df= pd.read_csv('data/train.csv')
df.head()
```

Out[2]:

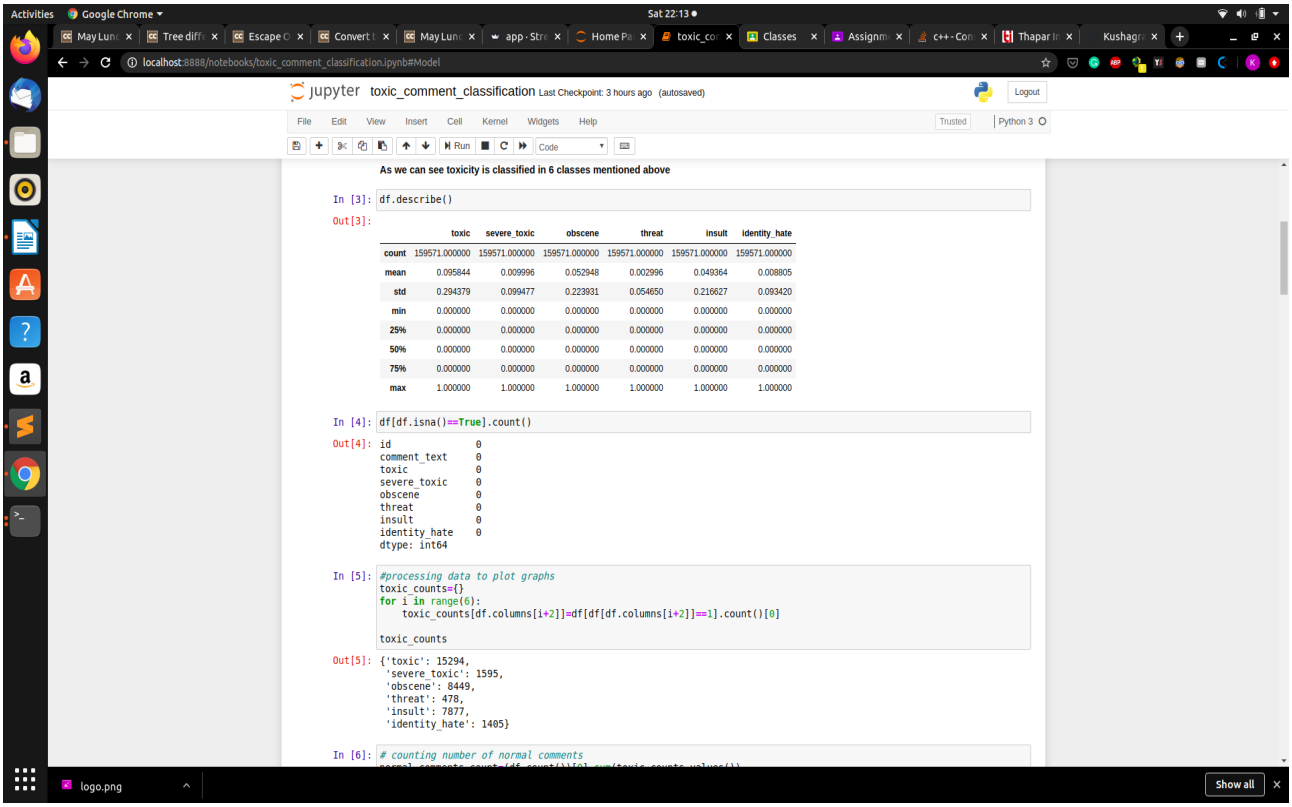
	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

As we can see toxicity is classified in 6 classes mentioned above

In [3]:
```
df.describe()
```

Out[3]:

	toxic	severe_toxic	obscene	threat	insult	identity_hate
count	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000	159571.000000
mean	0.095844	0.009996	0.052948	0.002996	0.049364	0.008805
std	0.294379	0.099477	0.223931	0.054650	0.216627	0.093420
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000



In [6]:
```
# counting number of normal comments
normal_comments_count=(df.count())[0]-sum(toxic_counts.values())
temp_dict={'normal':normal_comments_count}

chart_data= {**temp_dict,**toxic_counts}
chart_data
```

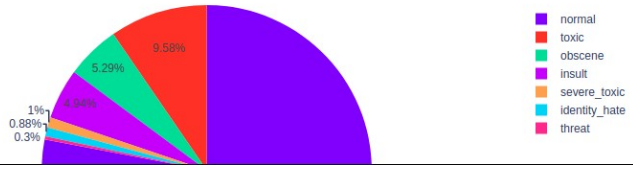
Out[6]:
```
{'normal': 124473,
 'toxic': 15294,
 'severe_toxic': 1595,
 'obscene': 8449,
 'threat': 478,
 'insult': 7877,
 'identity_hate': 1405}
```

In [7]:
```
toxic_counts
```

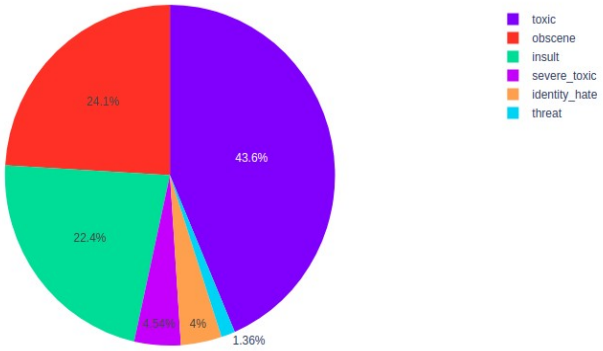
Out[7]:
```
{'toxic': 15294,
 'severe_toxic': 1595,
 'obscene': 8449,
 'threat': 478,
 'insult': 7877,
 'identity_hate': 1405}
```

Poltting

In [8]:
```
fig_total= go.Figure([go.Pie(labels=list(chart_data.keys()),values=list(chart_data.values()))])
# chart_data.values()
fig_total.show()
```



In [9]:
```
fig_toxic= go.Figure([go.Pie(labels=list(toxic_counts.keys()),values=list(toxic_counts.values()))])
fig_toxic.show()
```

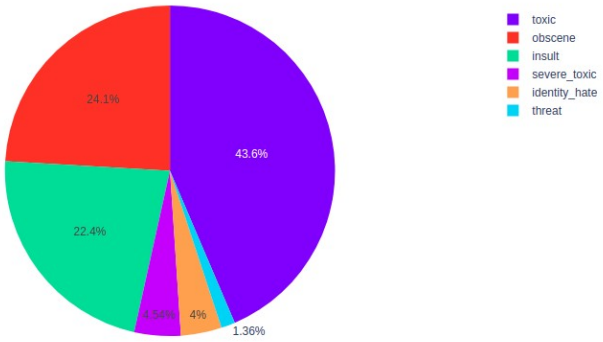


Preprocessing

In [10]:
```
classes=list(toxic_counts.keys())
y=df[classes].values
comments=df['comment_text']
```

In [11]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

In [9]:
```
fig_toxic= go.Figure([go.Pie(labels=list(toxic_counts.keys()),values=list(toxic_counts.values()))])
fig_toxic.show()
```



Preprocessing

In [10]:
```
classes=list(toxic_counts.keys())
y=df[classes].values
comments=df['comment_text']
```

In [11]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

Preprocessing

In [10]:
```
classes=list(toxic_counts.keys())
y=df[classes].values
comments=df['comment_text']
```

In [11]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
```

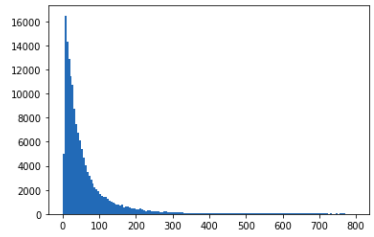
```
Using TensorFlow backend.
```

In [12]:
```
# Tokenization
max_features=20000
tokenizer= Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(comments))
tokenized_comments=tokenizer.texts_to_sequences(comments)
tokenized_comments[0]
```

Out[12]:
```
[688,
 75,
 1,
 126,
 130,
 177,
 29,
 672,
 4511,
 12052,
 1116,
 86,
 331,
 51,
 2278,
 11448,
 50,
 6864,
 15,
 60,
 2756,
 148,
 7,
 2937,
 34,
 117,
 1221
```

Plot to determine maximum length of sentence

In [13]:
```
#padding
total_words_per_sentence=[len(comment) for comment in tokenized_comments]
plt.hist(total_words_per_sentence,bins=np.arange(0,800,5))
plt.show()
```



There are negligible sentences with length greater than 200. thus we select max_length=200

In [14]:
```
#padding
max_length=200
X= pad_sequences(tokenized_comments,max_length)
X
```

Out[14]:
```
array([[    0,     0,     0, ...,  4583,  2273,   985],
       [    0,     0,     0, ...,   589,  8377,   182],
       [    0,     0,     0, ...,     1,   737,   468],
       ...,
       [    0,     0,     0, ...,  3509, 13675,  4528],
       [    0,     0,     0, ...,   151,    34,    11],
       [    0,     0,     0, ...,  1627,  2056,    88]], dtype=int32)
```

Building Deep Learning Model

In [15]:
```
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers import Bidirectional, GlobalMaxPool1D
from keras.models import Model
```

Building Deep Learning Model

In [15]:
```
from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation
from keras.layers import Bidirectional, GlobalMaxPool1D
from keras.models import Model
from keras import initializers, regularizers, constraints, optimizers, layers
```

In [16]:
```
#layers
model=Model()
input_layer=(Input(shape=(max_length,)))
embed_size=128 #hit and trial
embedded=(Embedding(max_features,embed_size))(input_layer)
LSTM_layer=(LSTM(60,return_sequences= True,name='lstm'))(embedded)
max_pool=(GlobalMaxPool1D())(LSTM_layer)
dropout=(Dropout(0.1))(max_pool)
final=(Dense(6,activation='sigmoid'))(dropout)
```

In [17]:
```
#compiling model
model=Model(inputs=input_layer,outputs=final)
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

```
WARNING:tensorflow:From /home/zues7744/.local/lib/python3.6/site-packages/tensorflow/python/ops/nn_impl.py:180: ad
d_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a
future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

Training

In [18]:
```
batch_size=16
epochs=3
model.fit(X,y,batch_size=batch_size,epochs=epochs,validation_split=0.3)
```

```
WARNING:tensorflow:From /home/zues7744/.local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:422:
The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 111699 samples, validate on 47872 samples
Epoch 1/3
111699/111699 [==============================] - 548s 5ms/step - loss: 0.0746 - accuracy: 0.9763 - val_loss: 0.049
1 - val_accuracy: 0.9821
Epoch 2/3
111699/111699 [==============================] - 542s 5ms/step - loss: 0.0452 - accuracy: 0.9833 - val_loss: 0.046
7 - val_accuracy: 0.9827
```

Training

In [18]:
```
batch_size=16
epochs=3
model.fit(X,y,batch_size=batch_size,epochs=epochs,validation_split=0.3)
```

```
WARNING:tensorflow:From /home/zues7744/.local/lib/python3.6/site-packages/keras/backend/tensorflow_backend.py:422:
The name tf.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 111699 samples, validate on 47872 samples
Epoch 1/3
111699/111699 [==============================] - 548s 5ms/step - loss: 0.0746 - accuracy: 0.9763 - val_loss: 0.049
1 - val_accuracy: 0.9821
Epoch 2/3
111699/111699 [==============================] - 542s 5ms/step - loss: 0.0452 - accuracy: 0.9833 - val_loss: 0.046
7 - val_accuracy: 0.9827
Epoch 3/3
111699/111699 [==============================] - 552s 5ms/step - loss: 0.0393 - accuracy: 0.9849 - val_loss: 0.046
4 - val_accuracy: 0.9828
```

Out[18]:
```
<keras.callbacks.callbacks.History at 0x7f52b2154828>
```

In [19]:
```
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 200)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 200, 128)          2560000
_________________________________________________________________
lstm (LSTM)                  (None, 200, 60)           45360
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 60)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 60)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 366
=================================================================
Total params: 2,605,726
Trainable params: 2,605,726
Non-trainable params: 0
_________________________________________________________________
```

Saving Model

```
111699/111699 [==============================] - 552s 5ms/step - loss: 0.0393 - accuracy: 0.9849 - val_loss: 0.0464 - val_accuracy: 0.9828
```

Out[18]:
```
<keras.callbacks.callbacks.History at 0x7f52b2154828>
```

In [19]:
```
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 200)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 200, 128)          2560000
_________________________________________________________________
lstm (LSTM)                  (None, 200, 60)           45360
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 60)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 60)                0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 366
=================================================================
Total params: 2,605,726
Trainable params: 2,605,726
Non-trainable params: 0
_________________________________________________________________
```

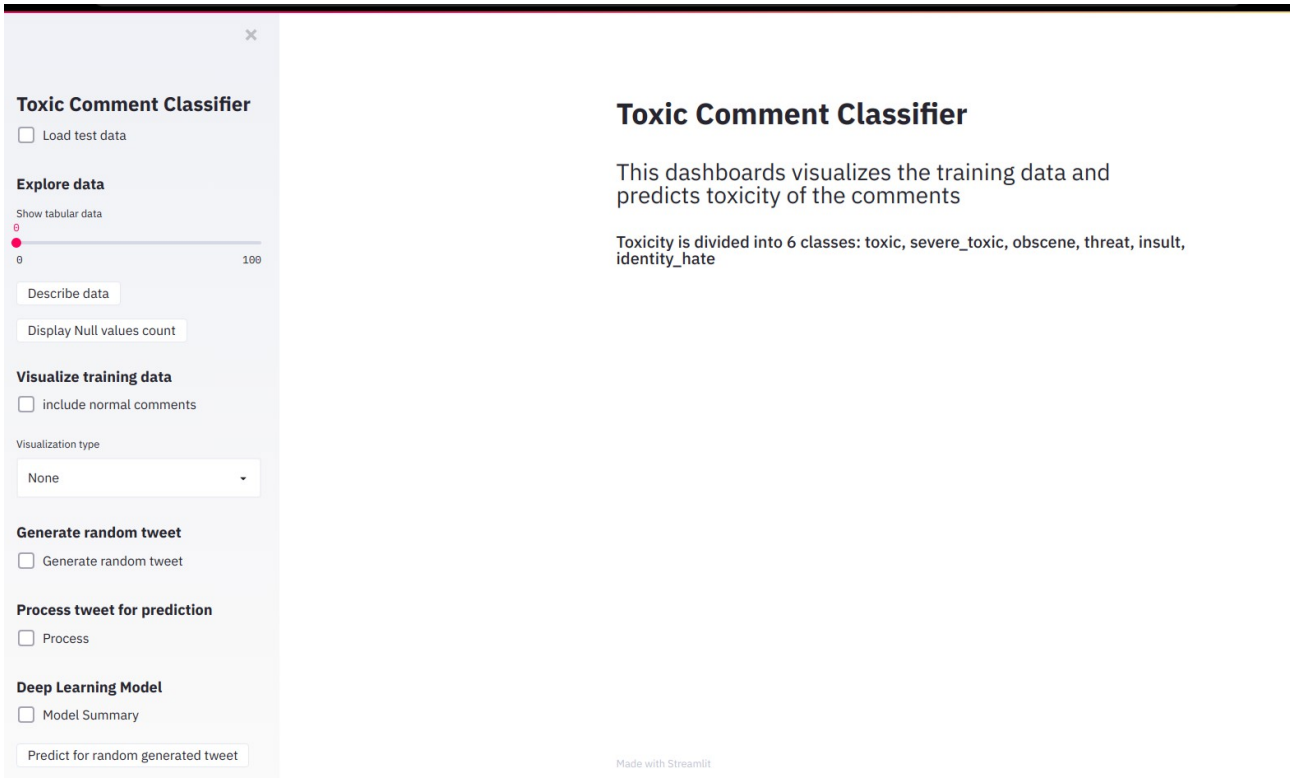
Saving Model ¶

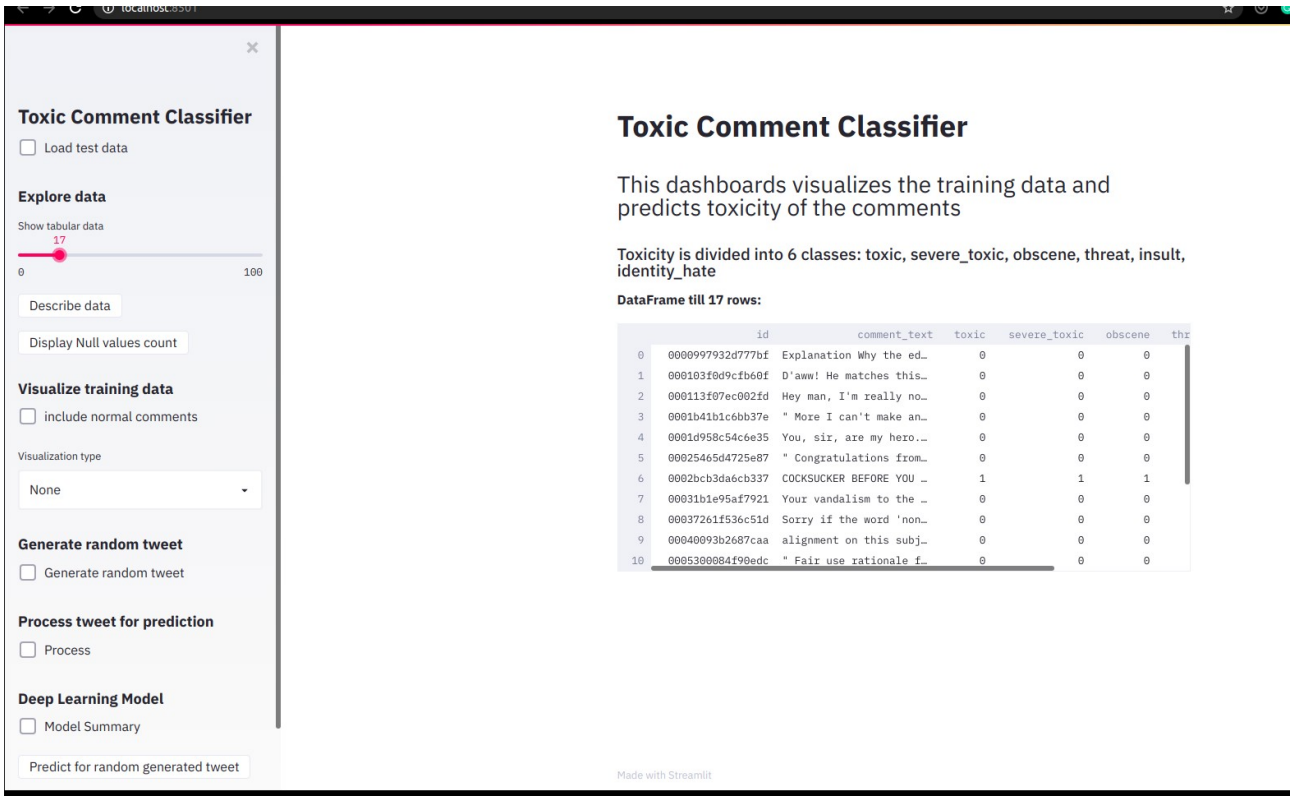
In [21]:
```
from keras.models import model_from_json
```

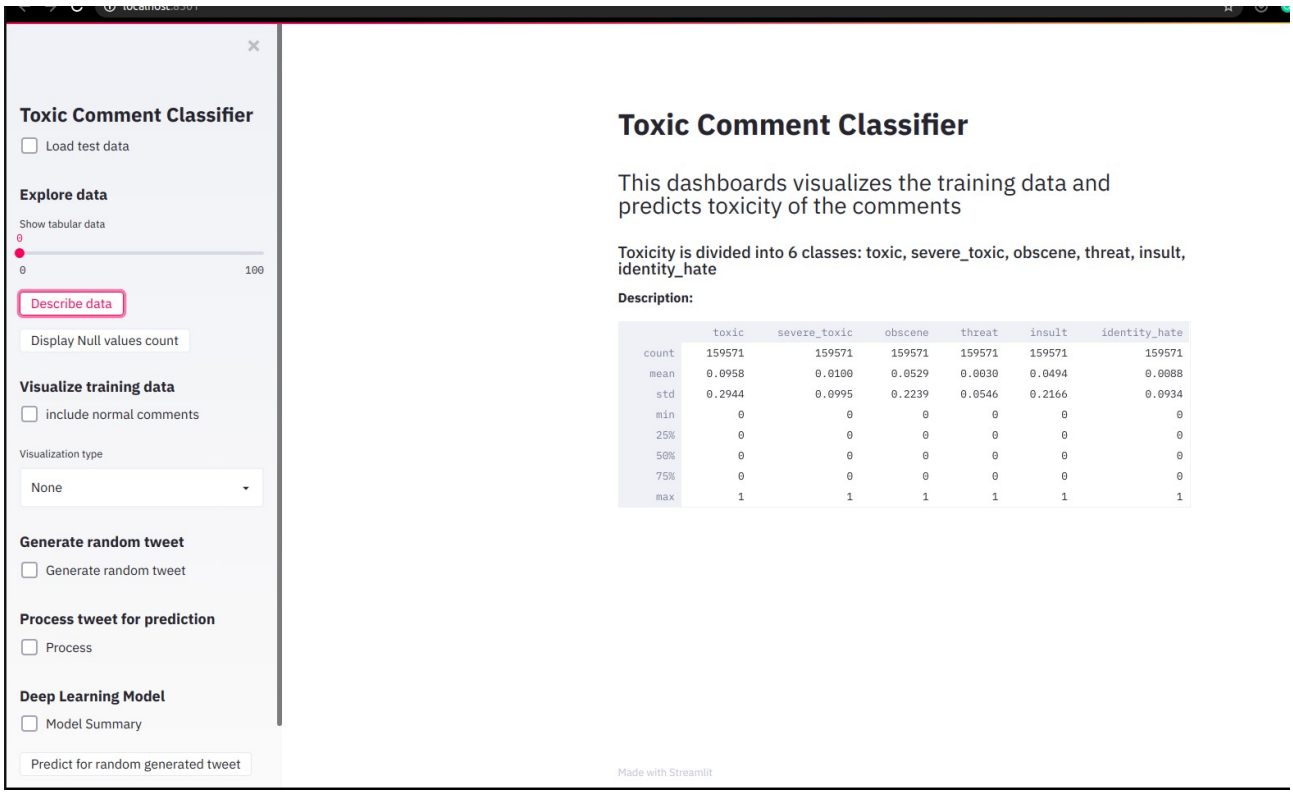
In [23]:
```
model_json=model.to_json()
with open("model.json","w") as json_file:
    json_file.write(model_json)

model.save_weights("weights.h5")
```

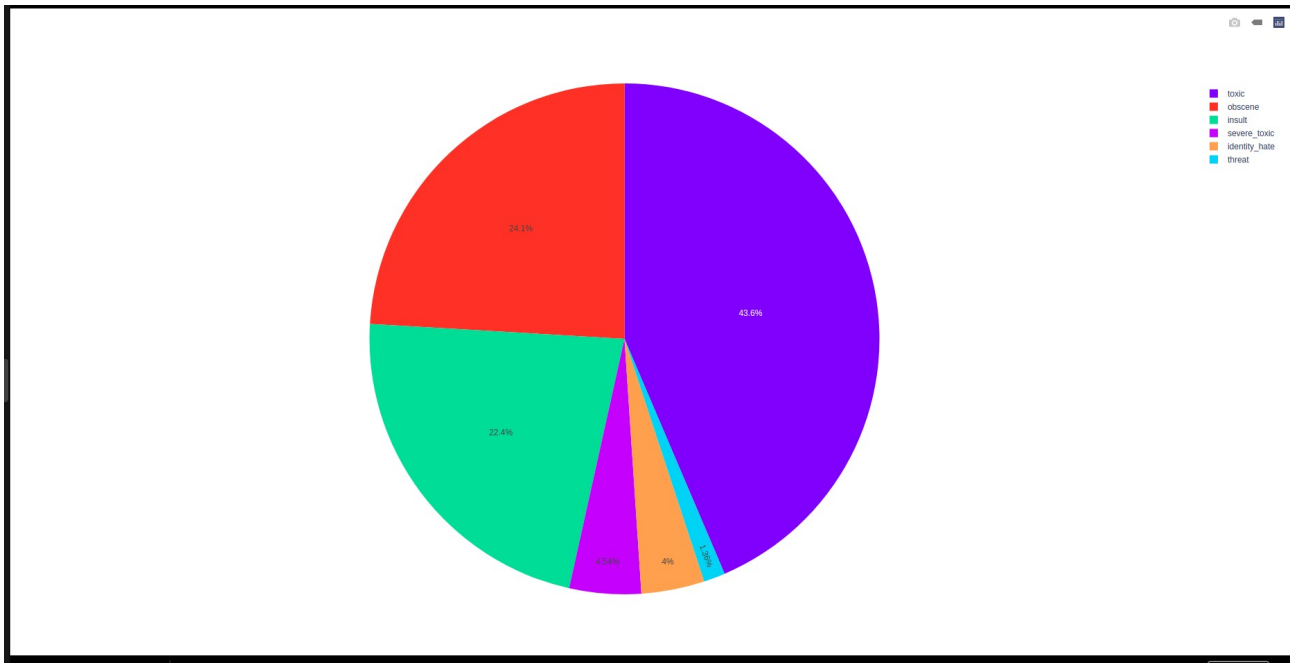
2) Dashboard

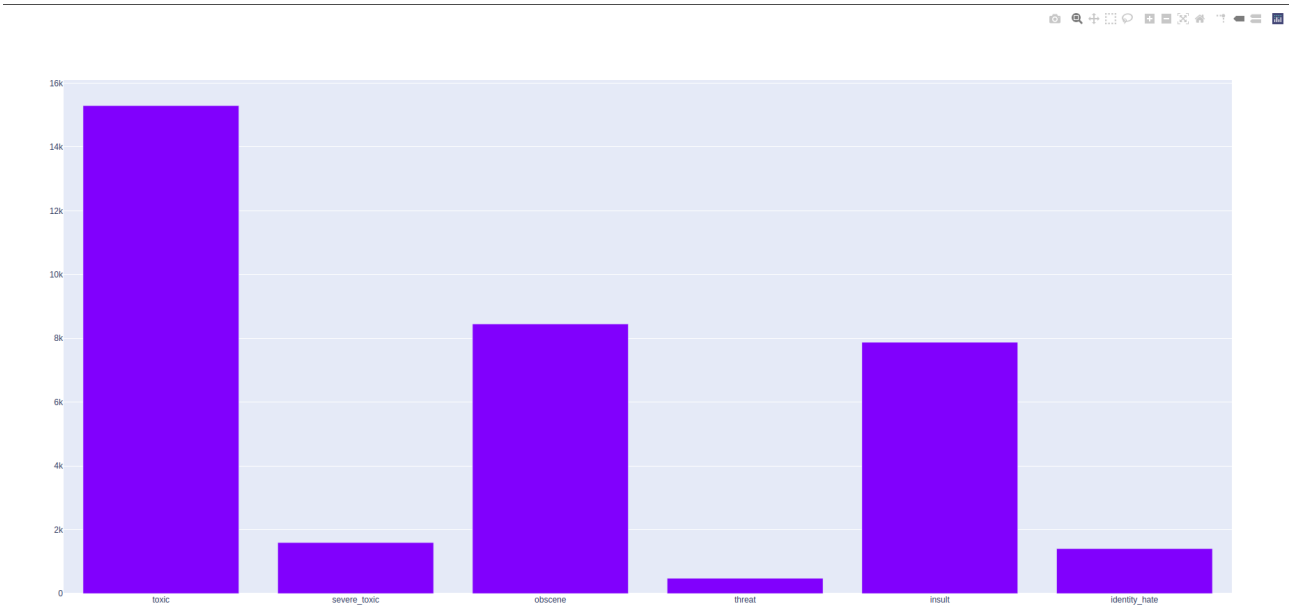


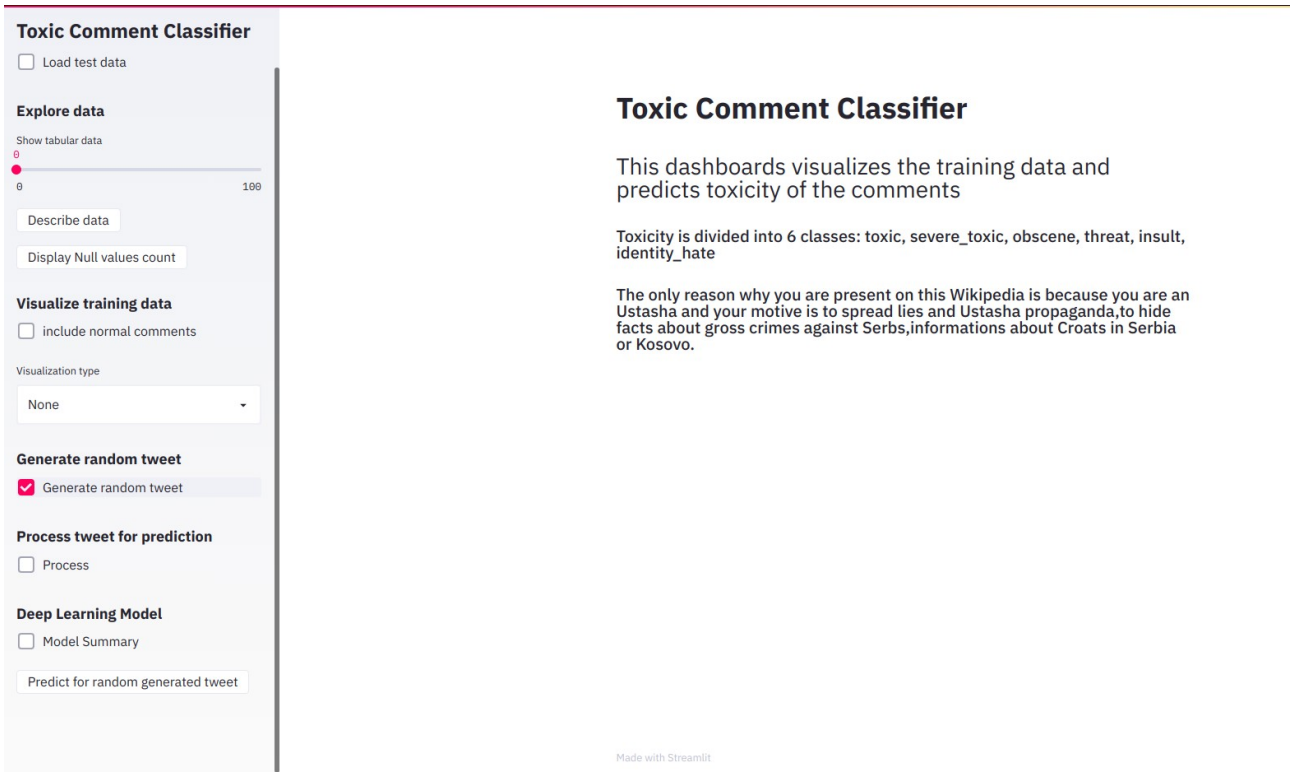

Toxic Comment Classifier
Load test data
Explore data
Show tabular data
17
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
DataFrame till 17 rows:
id comment_text toxic severe_toxic obscene thr
0 0000997932d777bf Explanation Why the ed… 0 0 0
1 000103f0d9cfb60f D'aww! He matches this… 0 0 0
2 000113f07ec002fd Hey man, I'm really no… 0 0 0
3 0001b41b1c6bb37e " More I can't make an… 0 0 0
4 0001d958c54c6e35 You, sir, are my hero.… 0 0 0
5 00025465d4725e87 " Congratulations from… 0 0 0
6 0002bcb3da6cb337 COCKSUCKER BEFORE YOU … 1 1 1
7 00031b1e95af7921 Your vandalism to the … 0 0 0
8 00037261f536c51d Sorry if the word 'non… 0 0 0
9 00040093b2687caa alignment on this subj… 0 0 0
10 0005300084f90edc " Fair use rationale f… 0 0 0
Made with Streamlit

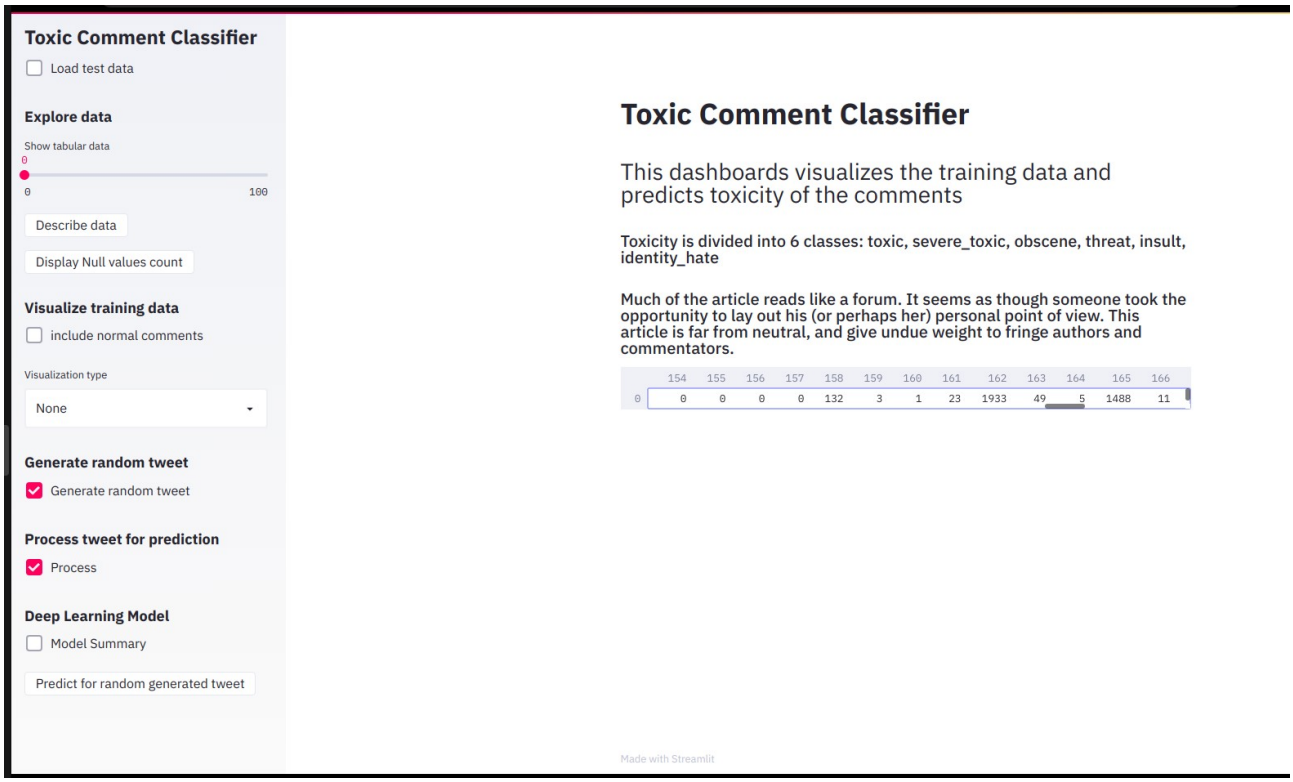

Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
Description:
toxic severe_toxic obscene threat insult identity_hate
count 159571 159571 159571 159571 159571 159571
mean 0.0958 0.0100 0.0529 0.0030 0.0494 0.0088
std 0.2944 0.0995 0.2239 0.0546 0.2166 0.0934
min 0 0 0 0 0 0
25% 0 0 0 0 0 0
50% 0 0 0 0 0 0
75% 0 0 0 0 0 0
max 1 1 1 1 1 1
Made with Streamlit

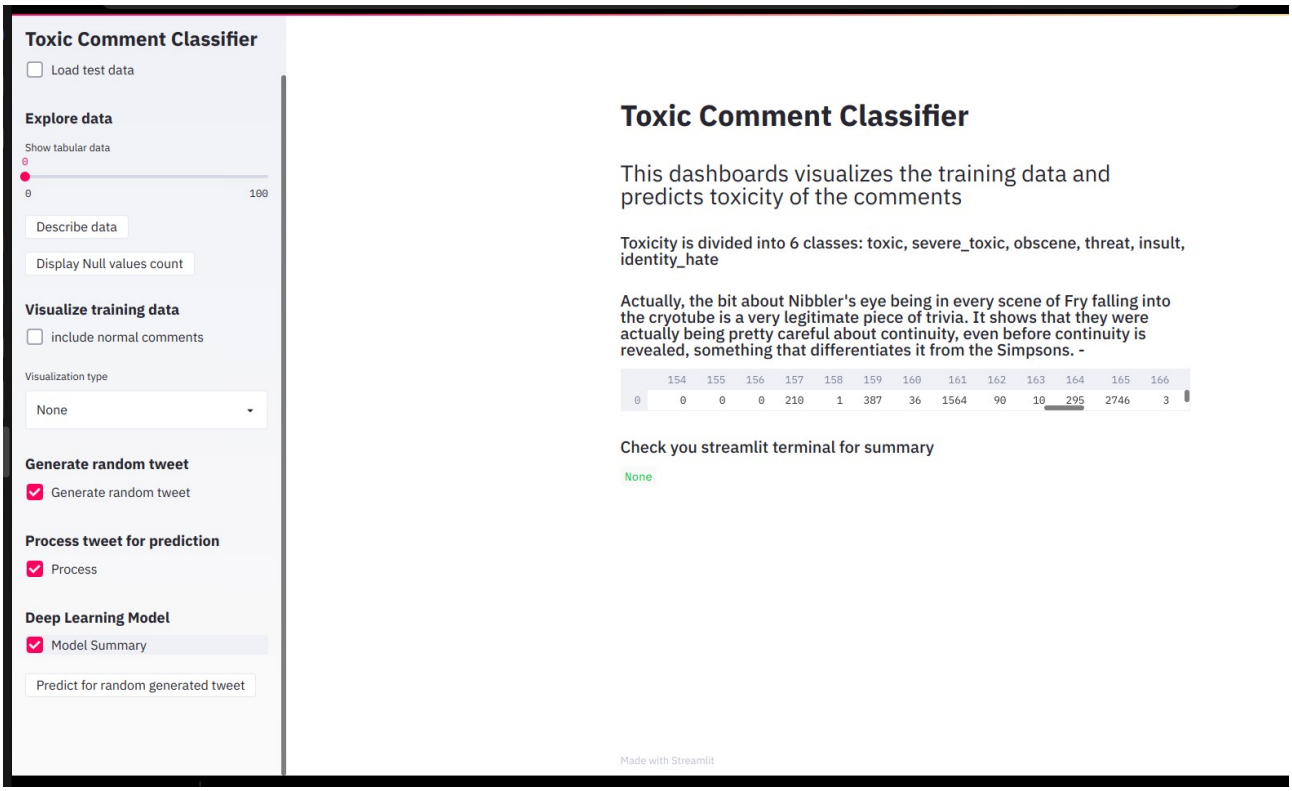

Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
Count of NULL values for all columns:
0
id 0
comment_text 0
toxic 0
severe_toxic 0
obscene 0
threat 0
insult 0
identity_hate 0
Made with Streamlit

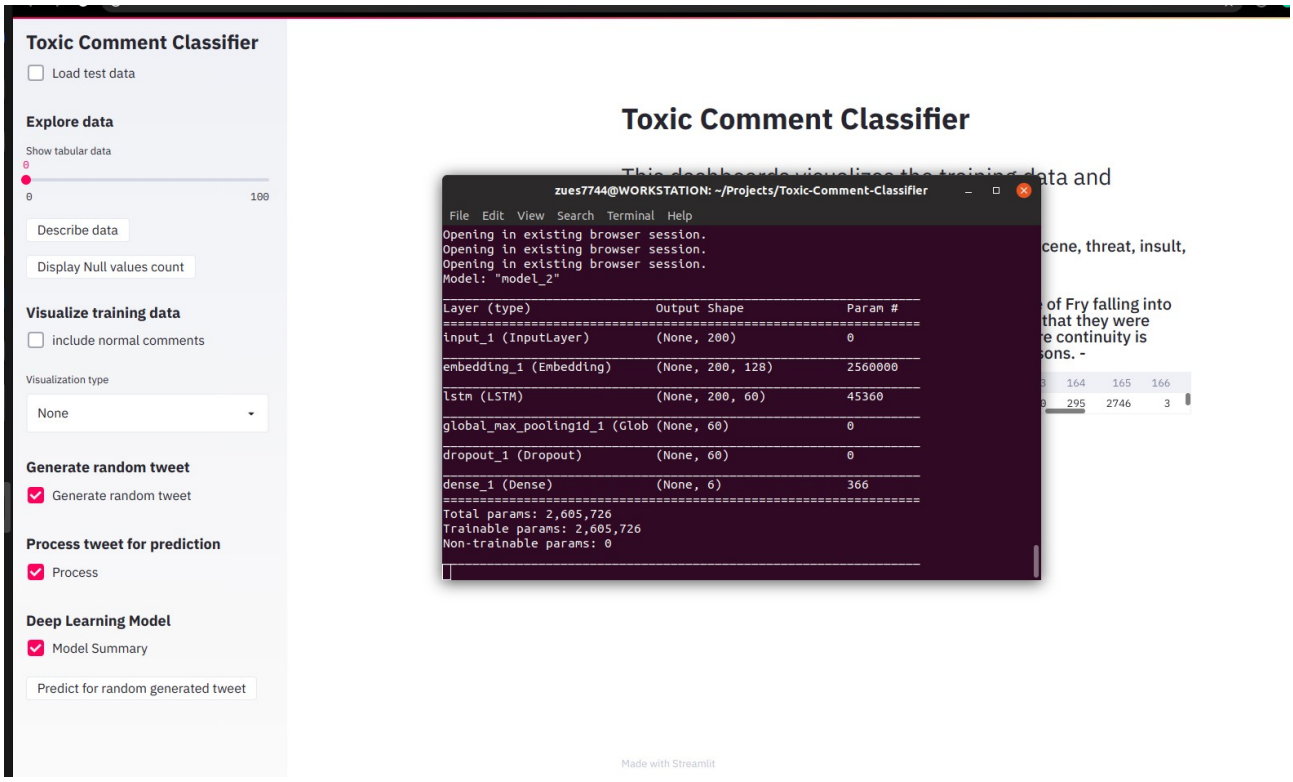

toxic
obscene
insult
severe_toxic
identity_hate
threat
43.6%
24.1%
22.4%
4%

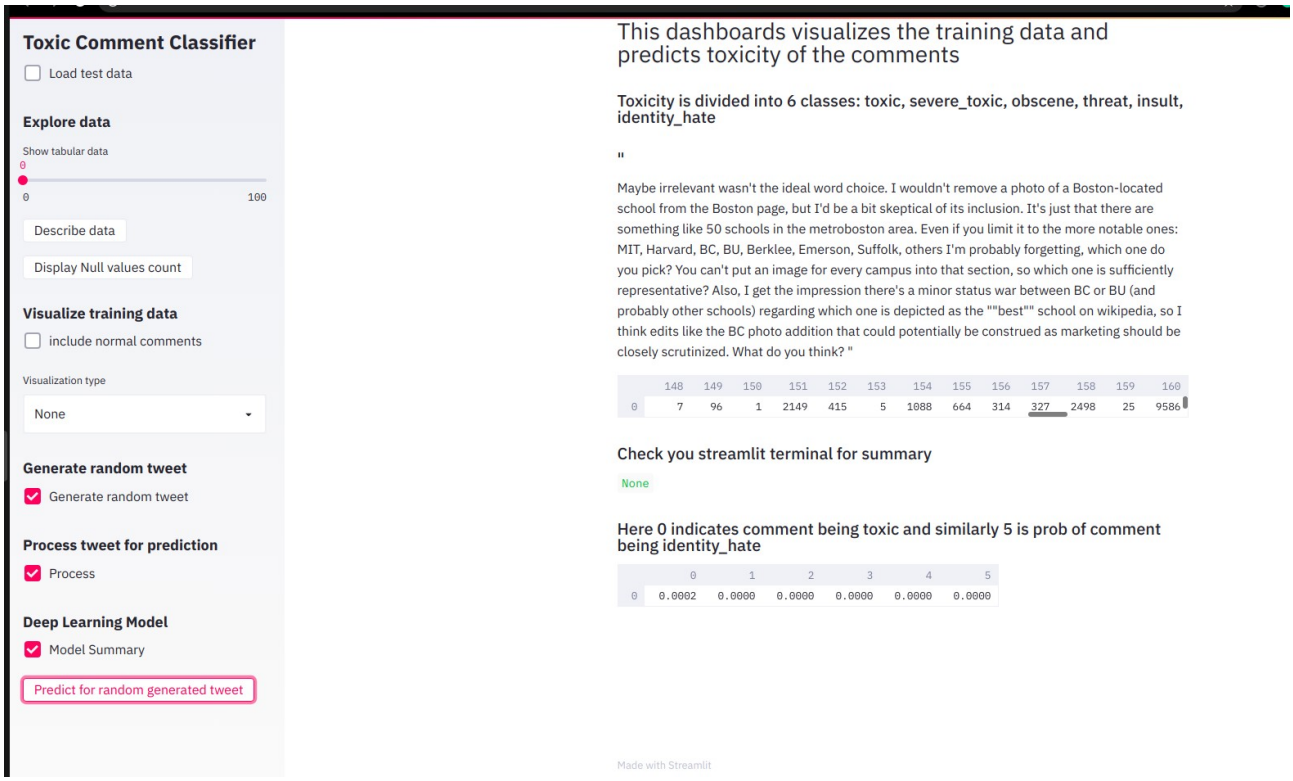

16k
14k
12k
10k
8k
6k
4k
2k
0
toxic
severe_toxic
obscene
threat
insult
identity_hate


Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
The only reason why you are present on this Wikipedia is because you are an Ustasha and your motive is to spread lies and Ustasha propaganda,to hide facts about gross crimes against Serbs,informations about Croats in Serbia or Kosovo.
Made with Streamlit


Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
Much of the article reads like a forum. It seems as though someone took the opportunity to lay out his (or perhaps her) personal point of view. This article is far from neutral, and give undue weight to fringe authors and commentators.
154 155 156 157 158 159 160 161 162 163 164 165 166
0 0 0 0 0 132 3 1 23 1933 49 5 1488 11
Made with Streamlit


Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
Actually, the bit about Nibbler's eye being in every scene of Fry falling into the cryotube is a very legitimate piece of trivia. It shows that they were actually being pretty careful about continuity, even before continuity is revealed, something that differentiates it from the Simpsons. -
154 155 156 157 158 159 160 161 162 163 164 165 166
0 0 0 0 210 1 387 36 1564 90 10 295 2746 3
Check you streamlit terminal for summary
None
Made with Streamlit


Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
Toxic Comment Classifier
zues7744@WORKSTATION: ~/Projects/Toxic-Comment-Classifier
File Edit View Search Terminal Help
Opening in existing browser session.
Opening in existing browser session.
Opening in existing browser session.
Model: "model_2"
Layer (type) Output Shape Param #
input_1 (InputLayer) (None, 200) 0
embedding_1 (Embedding) (None, 200, 128) 2560000
lstm (LSTM) (None, 200, 60) 45360
global_max_pooling1d_1 (Glob (None, 60) 0
dropout_1 (Dropout) (None, 60) 0
dense_1 (Dense) (None, 6) 366
Total params: 2,605,726
Trainable params: 2,605,726
Non-trainable params: 0
Made with Streamlit


Toxic Comment Classifier
Load test data
Explore data
Show tabular data
0
0
100
Describe data
Display Null values count
Visualize training data
include normal comments
Visualization type
None
Generate random tweet
Generate random tweet
Process tweet for prediction
Process
Deep Learning Model
Model Summary
Predict for random generated tweet
This dashboards visualizes the training data and predicts toxicity of the comments
Toxicity is divided into 6 classes: toxic, severe_toxic, obscene, threat, insult, identity_hate
"
Maybe irrelevant wasn't the ideal word choice. I wouldn't remove a photo of a Boston-located school from the Boston page, but I'd be a bit skeptical of its inclusion. It's just that there are something like 50 schools in the metroboston area. Even if you limit it to the more notable ones: MIT, Harvard, BC, BU, Berklee, Emerson, Suffolk, others I'm probably forgetting, which one do you pick? You can't put an image for every campus into that section, so which one is sufficiently representative? Also, I get the impression there's a minor status war between BC or BU (and probably other schools) regarding which one is depicted as the ""best"" school on wikipedia, so I think edits like the BC photo addition that could potentially be construed as marketing should be closely scrutinized. What do you think? "
148 149 150 151 152 153 154 155 156 157 158 159 160
0 7 96 1 2149 415 5 1088 664 314 327 2498 25 9586
Check you streamlit terminal for summary
None
Here 0 indicates comment being toxic and similarly 5 is prob of comment being identity_hate
0 1 2 3 4 5
0 0.0002 0.0000 0.0000 0.0000 0.0000 0.0000
Made with Streamlit