Description of the project

hello all the viewers who want to make a mchine learning model which identifies how toxic are the text/comments. and rate them according to the categories like threat, toxicity, insult, obscene language.

And build a Gradio Application to view the functioning of machine learning model

The process of building the application and presenting it

1. Installing the dependencies like Tensdorflow, Tensorflow -gpu pandas sklearn

```
1.   Tensorflow - for building the machine learning models with help keras of
2.   Tensorflow-gpu - for harnessing the power of gpu for better computation power
3.   Pandas - for building the data set from the given train CSV file
```

Pre Processing the data

first we have to perform tokenization of data which is done by using Textvectorization from tensorflow.keras.layers

And prepare datapipeline to fed into the deep learning model

Building a deep learning model

Creating a sequential model using tensorflow.keras.models importing Sequential Model

Making Model prediction

Evaluate the working of model

Making a gradio for displaying the functionality of the machine learning model

1 Installing required Dependencies.

```
!pip install tensorflow tensorflow-gpu pandas matplotlib sklearn
```

```
import os
import pandas as pd
import tensorflow as tf
import numpy as np
```

```
os.path.join('content','drive','MyDrive','Text toxicity model data','train.csv')  #/content/drive/MyDrive/Text toxicity model data/train
```

```
df = pd.read_csv('/content/drive/MyDrive/Text toxicity model data/train.csv')
```

```
df.head()
```

	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

```
df.tail()
```

```
# lets get a look at the coloumns
df.iloc[2]['comment_text']
```

```
'Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me through edits instead of my talk page. He seems to care more about the formatting than the actual info.'
```

Pre Procesing the data

```
from tensorflow.keras.layers import TextVectorization
```

```
TextVectorization??
# A preprocessing layer which maps text features to integer sequences.
```

```
df.columns
```

```
Index(['id', 'comment_text', 'toxic', 'severe_toxic', 'obscene', 'threat',
       'insult', 'identity_hate'],
      dtype='object')
```

```
df[df.columns[2:]].values
```

```
array([[0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       ...,
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0]])
```

```
X = df['comment_text']   # seperating the data into comment_text columns
y =df[df.columns[2:]].values #seperating the data into features columns
```

```
X
```

```
0         Explanation\nWhy the edits made under my usern...
1         D'aww! He matches this background colour I'm s...
2         Hey man, I'm really not trying to edit war. It...
3         "\nMore\nI can't make any real suggestions on ...
4         You, sir, are my hero. Any chance you remember...
                              ...
159566    ":::::And for the second time of asking, when ...
159567    You should be ashamed of yourself \n\nThat is ...
159568    Spitzer \n\nUmm, theres no actual article for ...
159569    And it looks like it was actually you who put ...
159570    "\nAnd ... I really don't think you understand...
Name: comment_text, Length: 159571, dtype: object
```

```
y
```

```
array([[0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       ...,
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0]])
```

```
MAX_WORDS = 300000  #number of words in vocab
```

```
vectorizer = TextVectorization(max_tokens = MAX_WORDS, output_sequence_length=2000, output_mode = 'int')
```

```
vectorizer.adapt(X.values)
```

```
vectorizer('hello world, life is great')
```

```
<tf.Tensor: shape=(2000,), dtype=int64, numpy=array([288, 263, 306, ...,   0,   0,   0])>
```

```
vectorizer.get_vocabulary()
```

```
['',
 '[UNK]',
 'the',
 'to',
```

```
'of',
'and',
'a',
'you',
'i',
'is',
'that',
'in',
'it',
'for',
'this',
'not',
'on',
'be',
'as',
'have',
'are',
'your',
'with',
'if',
'article',
'was',
'or',
'but',
'page',
'my',
'an',
'from',
'by',
'do',
'at',
'about',
'me',
'so',
'wikipedia',
'can',
'what',
'there',
'all',
'has',
'will',
'talk',
'please',
'would',
'its',
'no',
'one',
'just',
'like',
'they',
'he',
'dont',
'which',
'any'
```

```
vectorized_text = vectorizer(X.values)
```

```
vectorized_text
```

```
<tf.Tensor: shape=(159571, 2000), dtype=int64, numpy=
array([[   645,     76,      2, ...,      0,      0,      0],
       [219427,     54,   2489, ...,      0,      0,      0],
       [   425,    441,     70, ...,      0,      0,      0],
       ...,
       [ 32445,   7392,    383, ...,      0,      0,      0],
       [     5,     12,    534, ...,      0,      0,      0],
       [     5,      8,    130, ...,      0,      0,      0]])>
```

Building a data pipeline

```
# MAP -> Cache -> shuffle -> batch -> prefetch from tensor_slices, list_files

dataset = tf.data.Dataset.from_tensor_slices((vectorized_text, y))
dataset = dataset.cache()
dataset = dataset.shuffle(160000)
dataset = dataset.batch(16)
dataset = dataset.prefetch(8)
```

```
batch_X, batch_y = dataset.as_numpy_iterator().next()
```

```
train = dataset.take(int(len(dataset)*.7))
val = dataset.skip(int(len(dataset)*.7)).take(int(len(dataset)*.2))
test = dataset.skip(int(len(dataset)*.9)).take(int(len(dataset)*.1))
```

```
train_generator =train.as_numpy_iterator()
```

```
train_generator.next()
```

```
(array([[146179,  10613,      8, ...,      0,      0,      0],
        [   191,    288,      5, ...,      0,      0,      0],
        [     9,     12,    130, ...,      0,      0,      0],
        ...,
        [   793,  22724,      0, ...,      0,      0,      0],
        [     8,    105,      7, ...,      0,      0,      0],
        [    40,     25,  26737, ...,      0,      0,      0]]),
 array([[0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [1, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0]]))
```

2 DEEP learning Model

2.1 Creating a sequential Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dropout, Bidirectional, Dense, Embedding
```

```
model = Sequential()
# Create the embedding layer
model.add(Embedding(MAX_WORDS+1,32))
# Bidirectional LSTM Layer
model.add(Bidirectional(LSTM(32, activation='tanh')))
# Feature extractor Fully connected layers
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
# Final layer
model.add(Dense(6, activation='sigmoid'))
```

```
model.compile(loss='BinaryCrossentropy', optimizer='Adam')
```

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, None, 32)          9600032

 bidirectional_2 (Bidirectio  (None, 64)               16640
 nal)

 dense_8 (Dense)             (None, 128)               8320

 dense_9 (Dense)             (None, 256)               33024

 dense_10 (Dense)            (None, 128)               32896

 dense_11 (Dense)            (None, 6)                 774

=================================================================
Total params: 9,691,686
Trainable params: 9,691,686
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(train, epochs=5, validation_data=val)
```

```
Epoch 1/5
6981/6981 [==============================] - 709s 101ms/step - loss: 0.0619 - val_loss: 0.0463
Epoch 2/5
6981/6981 [==============================] - 712s 102ms/step - loss: 0.0453 - val_loss: 0.0388
Epoch 3/5
6981/6981 [==============================] - 713s 102ms/step - loss: 0.0405 - val_loss: 0.0339
Epoch 4/5
6981/6981 [==============================] - 725s 104ms/step - loss: 0.0356 - val_loss: 0.0309
Epoch 5/5
6981/6981 [==============================] - 720s 103ms/step - loss: 0.0313 - val_loss: 0.0278
```

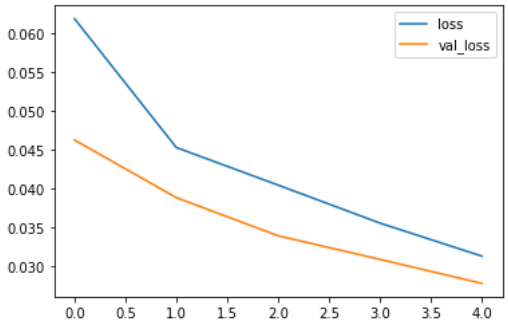
```
history.history
```

```
{'loss': [0.06187304109334946,
  0.045293889939785004,
  0.04045658931136131,
  0.03556494042277336,
  0.03131736069917679],
 'val_loss': [0.04626167565584183,
  0.0388483926653862,
  0.0339345782995224,
  0.030896401032805443,
  0.027813147753477097]}
```

```
from matplotlib import pyplot as plt
```

```
plt.figure(figsize=(8,5))
pd.DataFrame(history.history).plot()
plt.show()
```

```
<Figure size 576x360 with 0 Axes>
```



4 Making Model predictions

```
input_text = vectorizer('You freaking suck! I am going to hit you.')
```

```
np.expand_dims(input_text,0)
```

```
array([[   7, 7158,  397, ...,    0,    0,    0]])
```

```
batch = test.as_numpy_iterator().next()
```

```
batch_X, batch_y =test.as_numpy_iterator().next()
```

```
((model.predict(batch_X))>0.5).astype(int)
```

```
1/1 [==============================] - 0s 74ms/step
array([[0, 0, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
```

```
            [0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0],
            [0, 0, 0, 0, 0, 0]])
```

```
res = model.predict(np.expand_dims(input_text,0))
```

```
1/1 [==============================] - 0s 67ms/step
```

5 Evaluate Model

```
from tensorflow.keras.metrics import Precision, Recall, CategoricalAccuracy
```

```
pre = Precision()
re = Recall()
acc = CategoricalAccuracy()
```

```
for batch in test.as_numpy_iterator():
    # Unpack the batch
    X_true, y_true = batch
    # Make a prediction
    yhat = model.predict(X_true)

    # Flatten the predictions
    y_true = y_true.flatten()
    yhat = yhat.flatten()

    pre.update_state(y_true, yhat)
    re.update_state(y_true, yhat)
    acc.update_state(y_true, yhat)
```

```
1/1 [==============================] - 0s 71ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 75ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 67ms/step
1/1 [==============================] - 0s 68ms/step
1/1 [==============================] - 0s 70ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 64ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 65ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 60ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 59ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 61ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 56ms/step
1/1 [==============================] - 0s 57ms/step
```

```
1/1 [==============================] - 0s 66ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 62ms/step
1/1 [==============================] - 0s 58ms/step
1/1 [==============================] - 0s 66ms/step
```

```
print(f'Precision: {pre.result().numpy()}, Recall:{re.result().numpy()}, Accuracy:{acc.result().numpy()}')
```

```
Precision: 0.9011370539665222, Recall:0.7827160358428955, Accuracy:0.4874624013900757
```

6 Test and Gradio App

```
!pip install gradio jinja2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting gradio
  Downloading gradio-3.14.0-py3-none-any.whl (13.8 MB)
     |████████████████████████████████| 13.8 MB 4.6 MB/s
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (2.11.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from gradio) (1.21.6)
Collecting uvicorn
  Downloading uvicorn-0.20.0-py3-none-any.whl (56 kB)
     |████████████████████████████████| 56 kB 6.0 MB/s
Collecting ffmpy
  Downloading ffmpy-0.3.0.tar.gz (4.8 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from gradio) (1.3.5)
Collecting fastapi
  Downloading fastapi-0.88.0-py3-none-any.whl (55 kB)
     |████████████████████████████████| 55 kB 4.3 MB/s
Requirement already satisfied: markupsafe in /usr/local/lib/python3.8/dist-packages (from gradio) (2.0.1)
Collecting orjson
  Downloading orjson-3.8.3-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (278 kB)
     |████████████████████████████████| 278 kB 80.4 MB/s
Requirement already satisfied: aiohttp in /usr/local/lib/python3.8/dist-packages (from gradio) (3.8.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.8/dist-packages (from gradio) (3.2.2)
Requirement already satisfied: pydantic in /usr/local/lib/python3.8/dist-packages (from gradio) (1.10.2)
Collecting python-multipart
  Downloading python-multipart-0.0.5.tar.gz (32 kB)
Requirement already satisfied: altair in /usr/local/lib/python3.8/dist-packages (from gradio) (4.2.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.8/dist-packages (from gradio) (6.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.8/dist-packages (from gradio) (2022.11.0)
Collecting markdown-it-py[linkify,plugins]
  Downloading markdown_it_py-2.1.0-py3-none-any.whl (84 kB)
     |████████████████████████████████| 84 kB 4.2 MB/s
Requirement already satisfied: pillow in /usr/local/lib/python3.8/dist-packages (from gradio) (7.1.2)
Collecting websockets>=10.0
  Downloading websockets-10.4-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1
     |████████████████████████████████| 106 kB 63.9 MB/s
Collecting pycryptodome
  Downloading pycryptodome-3.16.0-cp35-abi3-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.wh
     |████████████████████████████████| 2.3 MB 58.6 MB/s
Collecting pydub
  Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Collecting httpx
  Downloading httpx-0.23.1-py3-none-any.whl (84 kB)
     |████████████████████████████████| 84 kB 5.0 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from gradio) (2.23.0)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (4.0
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (6.0.3)
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (2.
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (1.3.3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (1.8.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (22.1.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.8/dist-packages (from aiohttp->gradio) (1.3.1)
Requirement already satisfied: idna>=2.0 in /usr/local/lib/python3.8/dist-packages (from yarl<2.0,>=1.0->aiohttp->gradio) (2.10)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.8/dist-packages (from altair->gradio) (0.4)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.8/dist-packages (from altair->gradio) (4.3.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.8/dist-packages (from altair->gradio) (0.12.0)
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=3.0->altai
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.8/dist-packages (from jso
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.8/dist-packages (from importlib-resources>=1.4.0->jsonschem
```

```
import tensorflow as tf
import gradio as gr
```

```
model.save('text_toxicity.h5')
```

```
model = tf.keras.models.load_model('text_toxicity.h5')
```

```
input_str = vectorizer('hey i freaken hate you!')
```

```
res = model.predict(np.expand_dims(input_str,0))
```

```
1/1 [==============================] - 1s 716ms/step
```

```
res
```

```
array([[0.8268359 , 0.00182657, 0.02714728, 0.01612768, 0.2287659 ,
        0.05538357]], dtype=float32)
```

```
def score_text(Text):
    vectorized_comment = vectorizer([Text])
    results = model.predict(vectorized_comment)

    text = ''
    for idx, col in enumerate(df.columns[2:]):
        text += '{}: {}\n'.format(col, results[0][idx]>0.5)

    return text
```

```
interface = gr.Interface(fn=score_text,
                         inputs=gr.inputs.Textbox(lines=2, placeholder='text to score'),
                        outputs='text')
```

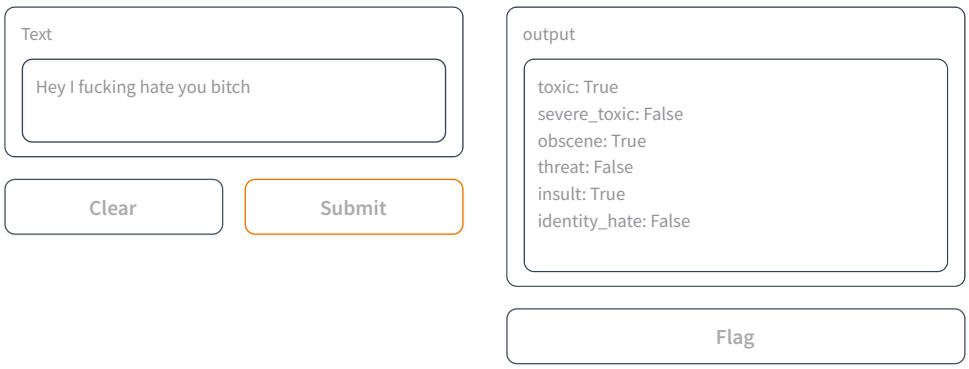
```
/usr/local/lib/python3.8/dist-packages/gradio/inputs.py:26: UserWarning: Usage of gradio.inputs is deprecated, and will not be supp
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `optional` parameter is deprecated, and it has no eff
  warnings.warn(value)
/usr/local/lib/python3.8/dist-packages/gradio/deprecation.py:40: UserWarning: `numeric` parameter is deprecated, and it has no effe
  warnings.warn(value)
```

```
interface.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`

Setting up a public link... we have recently upgraded the way public links are generated. If you encounter any problems, please rep
.
Running on public URL: https://3c8e6bb1-5f03-4673.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades (NEW!), check out Spaces: https://huggingface.co/
```



Use via API · Built with Gradio

✓ 2s completed at 11:38 PM