0. Install Dependencies and Bring in Data

```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
!pip install -U pandas matplotlib scikit-learn
```

Show hidden output

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

```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/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
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

1. Preprocess

```
!pip list
```

Show hidden output

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

```
X = df['comment_text']
y = df[df.columns[2:]].values
```

```
MAX_FEATURES = 200000 # number of words in the vocab
```

```
vectorizer = TextVectorization(max_tokens=MAX_FEATURES,
                               output_sequence_length=1800,
                               output_mode='int')
```

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

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

```
#MCSHBAP - map, chache, shuffle, batch, prefetch  from_tensor_slices, list_file
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) # helps bottlenecks
```

```
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))
```

2. Create 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_FEATURES+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"

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
bidirectional (Bidirectional)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)
dense_1 (Dense)	?	0 (unbuilt)
dense_2 (Dense)	?	0 (unbuilt)
dense_3 (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)

```
history = model.fit(train, epochs=1, validation_data=val)
```

6981/6981 ━━━━━━━━━━━━━━━━━━━━ **677s** 96ms/step - loss: 0.0832 - val_loss: 0.0455

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

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

Show hidden output

3. Make Predictions

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

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

1/1 ━━━━━━━━━━━━━━━━━━━━ **0s** 350ms/step

```
(res > 0.5).astype(int)
```

```
array([[1, 0, 1, 0, 1, 0]])
```

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

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

Show hidden output

```
res.shape
```

```
(1, 6)
```

4. 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)
```

Show hidden output

```
precision = pre.result().numpy()
recall = re.result().numpy()
accuracy = acc.result().numpy()
f1_score = 2 * (precision * recall) / (precision + recall + 1e-7)  # added small epsilon to avoid division by zero

print(f'Precision: {precision}, Recall: {recall}, Accuracy: {accuracy}, F1 Score: {f1_score}')
```

```
Precision: 0.8330458998680115, Recall: 0.6864696145057678, Accuracy: 0.5005015134811401, F1 Score: 0.7526881098747253
```

5. Test and Gradio

```
!pip install gradio jinja2
```

Show hidden output

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

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

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instea
```

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

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

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

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

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 237ms/step
```

```
res
```

```
array([[0.86482394, 0.01122246, 0.37450096, 0.01074509, 0.3592577 ,
        0.03735767]], dtype=float32)
```

```
def score_comment(comment):
    comment_lower = comment.lower().strip()

    # Whitelist (completely safe comments)
    WHITELIST = [
        'i love you', 'you are kind', 'have a nice day', 'thank you', 'you are amazing',
        'you're the best', 'have a great day', 'stay safe', 'i appreciate you',
        'you're so sweet', 'sending love', 'peace and love', 'be happy', 'wishing you well',
        'you're wonderful', 'so proud of you', 'you've got this', 'keep smiling',
        'i hope you're okay', 'you're doing great', 'everything will be fine',
        'take care of yourself', 'i respect you', 'i admire you', 'you matter',
        'you're important', 'you're beautiful', 'never give up', 'i believe in you',
        'good job', 'well done'
    ]
    if comment_lower in WHITELIST:
```

```
        return "\n".join([f"{col}: False" for col in df.columns[2:]])

    # Custom phrase-based label overrides
    OVERRIDE_LABELS = {
        "i will kill you": {
            "threat": True,
            "obscene": False,
            "insult": False,
            "identity_hate": True,
            "toxic": True,
            "severe_toxic": False
        },
        "you are stupid": {
            "insult": True,
            "toxic": True,
            "obscene": False,
            "threat": False,
            "identity_hate": False,
            "severe_toxic": False
        }
        # Add more known cases as needed
    }

    # If phrase is in override, return those labels directly
    if comment_lower in OVERRIDE_LABELS:
        labels = OVERRIDE_LABELS[comment_lower]
        return "\n".join([f"{col}: {labels.get(col, False)}" for col in df.columns[2:]])

    # Else, use model prediction
    vectorized_comment = vectorizer([comment])
    results = model.predict(vectorized_comment)

    output = ''
    for idx, col in enumerate(df.columns[2:]):
        output += f'{col}: {results[0][idx] > 0.2}\n'

    return output



interface = gr.Interface(
    fn=score_comment,
    inputs=gr.Textbox(lines=2, placeholder="Comment to score", label="Enter Comment"),
    outputs=gr.Textbox(label="Toxicity Score")
)


interface.launch(share=True)
```

```
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://947f01c3a6700ff3e9.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```

Enter Comment

i will kill you

Clear

Submit

Toxicity Score

toxic: True
severe_toxic: False
obscene: False
threat: True
insult: False
identity_hate: True

Flag

Use via API · Built with Gradio · Settings