

## ✓ Project Name:

## ✓ Toxicity Detector: Making Online Conversations Safer 🚨

Presented by : Balaji Ennawar

Project Type: Individual project

## ✓ Project Summary:

This project is all about making online spaces like social media and forums much safer and more pleasant. We've built a smart system called the Toxicity Detector that can automatically spot and flag harmful comments (like insults, threats, or hate speech) in real-time. It uses advanced computer intelligence to analyze text and predict how likely a comment is to be toxic, helping moderators keep online discussions healthy and respectful.

## ✓ GitHub Link

Double-click (or enter) to edit

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout, SpatialDropout1D
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
import os

# --- Task 1: Data Exploration and Preparation ---
print("--- Task 1: Data Exploration and Preparation ---")

# Load datasets
try:
    train_df = pd.read_csv('train.csv')
    test_df = pd.read_csv('test.csv')
    print("Datasets loaded successfully! ✅")
except FileNotFoundError:
    print("Error: 'train.csv' or 'test.csv' not found.")
    print("Please ensure both CSV files are uploaded to your Colab environment.")
    exit()

# Display first few rows of training data
print("\nFirst 5 rows of training data:")
print(train_df.head())

# Display info and check for missing values in training data
print("\nTraining data info:")
train_df.info()
print("\nMissing values in training data:")
print(train_df.isnull().sum())

# Fill missing comment_text with empty string
train_df['comment_text'].fillna('', inplace=True)
test_df['comment_text'].fillna('', inplace=True)
print("\nMissing 'comment_text' filled with empty strings.")

# Define target labels
LABEL_COLUMNS = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
print(f"\nTarget labels for classification: {LABEL_COLUMNS}")
```

```

# Prepare data for deep learning
max_features = 10000 # FURTHER REDUCED: Max number of words to keep in the vocabulary (from 20000 to 10000)
maxlen = 100         # Max length of sequences (comments)

# Initialize tokenizer
tokenizer = Tokenizer(num_words=max_features, oov_token="<unk>")
# Fit tokenizer on training comments
tokenizer.fit_on_texts(train_df['comment_text'])

# Convert text to sequences of integers
X_train = tokenizer.texts_to_sequences(train_df['comment_text'])
X_test = tokenizer.texts_to_sequences(test_df['comment_text'])

# Pad sequences to ensure uniform length
X_train = pad_sequences(X_train, maxlen=maxlen)
X_test = pad_sequences(X_test, maxlen=maxlen)
print(f"\nText data vectorized and padded. Sequence length: {maxlen}")

# Prepare labels (target variables)
y_train = train_df[LABEL_COLUMNS].values
# Note: For test_df, we don't have true labels for evaluation, so we'll use a dummy for structure
# In a real scenario, test_df would be for submission, or you'd split train_df for validation.
# For evaluation in this script, we'll split X_train and y_train further.

# Split training data for model validation
X_train_split, X_val_split, y_train_split, y_val_split = train_test_split(
    X_train, y_train, test_size=0.2, random_state=42
)
print(f"\nTraining data split: Train {X_train_split.shape}, Validation {X_val_split.shape}")

# --- Task 2: Model Development ---
print("\n--- Task 2: Model Development ---")

# Model parameters
embedding_dim = 64 # REDUCED: Embedding dimension (from 128 to 64)

# Build the Deep Learning Model (LSTM)
model = Sequential([
    Embedding(max_features, embedding_dim, input_length=maxlen),
    SpatialDropout1D(0.2), # Dropout layer to prevent overfitting
    LSTM(64, dropout=0.2, recurrent_dropout=0.2), # LSTM layer with dropout
    Dense(len(LABEL_COLUMNS), activation='sigmoid') # Output layer for multi-label classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()

# Callbacks for training
# EarlyStopping: Stop training if validation loss doesn't improve for 'patience' epochs
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
# ModelCheckpoint: Save the best model based on validation loss
model_checkpoint = ModelCheckpoint('toxicity_model.h5', monitor='val_loss', save_best_only=True)

# Train the model
epochs = 5 # Keep epochs low for demonstration; increase for better performance
print(f"\nStarting model training for {epochs} epochs...")
history = model.fit(
    X_train_split, y_train_split,
    epochs=epochs,
    batch_size=32,
    validation_data=(X_val_split, y_val_split),
    callbacks=[early_stopping, model_checkpoint]
)
print("\nModel training complete! 💎")

# Save the trained model and tokenizer
model.save('toxicity_model.h5')
with open('tokenizer.pkl', 'wb') as f:
    pickle.dump(tokenizer, f)
print("\nTrained model ('toxicity_model.h5') and tokenizer ('tokenizer.pkl') saved.")

# --- Task 3: Model Evaluation and Optimization ---
print("\n--- Task 3: Model Evaluation and Optimization ---")

# Load the best saved model for evaluation
best_model = load_model('toxicity_model.h5')

# Make predictions on the validation set
y_pred_probs = best_model.predict(X_val_split)
# Convert probabilities to binary predictions (threshold 0.5)
y_pred_binary = (y_pred_probs > 0.5).astype(int)

```

```
# Evaluate performance for each label
print("\n--- Classification Report for Each Toxicity Label ---")
for i, label in enumerate(LABEL_COLUMNS):
    print(f"\n--- Label: {label.upper()} ---")
    print(classification_report(y_val_split[:, i], y_pred_binary[:, i], zero_division=0))
    # Calculate ROC AUC score for each label
    roc_auc = roc_auc_score(y_val_split[:, i], y_pred_probs[:, i])
    print(f"ROC AUC Score for {label}: {roc_auc:.4f}")


# Visualize training history (accuracy and loss)
print("\n--- Visualizing Training History ---")
plt.figure(figsize=(12, 5))

# Plot training & validation accuracy values
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()

print("\n--- Colab script finished. Please download 'toxicity_model.h5' and 'tokenizer.pkl' ---")
```

--- Task 1: Data Exploration and Preparation ---  
 Datasets loaded successfully! 

First 5 rows of training data:

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

	severe_toxic	obscene	threat	insult	identity_hate
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

Training data info:

<class 'pandas.core.frame.DataFrame'>  
 RangeIndex: 159571 entries, 0 to 159570

Data columns (total 8 columns):

#	Column	Non-Null Count	Dtype
0	id	159571 non-null	object
1	comment_text	159571 non-null	object
2	toxic	159571 non-null	int64
3	severe_toxic	159571 non-null	int64
4	obscene	159571 non-null	int64
5	threat	159571 non-null	int64
6	insult	159571 non-null	int64
7	identity_hate	159571 non-null	int64

dtypes: int64(6), object(2)

memory usage: 9.7+ MB

Missing values in training data:

id	0
comment_text	0
toxic	0
severe_toxic	0
obscene	0
threat	0
insult	0
identity_hate	0

dtype: int64

Missing 'comment\_text' filled with empty strings.

Target labels for classification: ['toxic', 'severe\_toxic', 'obscene', 'threat', 'insult', 'identity\_hate']

/tmp/ipython-input-3778142015.py:40: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained  
 The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are settin

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[c

train\_df['comment\_text'].fillna('', inplace=True)

/tmp/ipython-input-3778142015.py:41: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained  
 The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are settin

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[c

test\_df['comment\_text'].fillna('', inplace=True)

Text data vectorized and padded. Sequence length: 100

Training data split: Train (127656, 100), Validation (31915, 100)

--- Task 2: Model Development ---

/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input\_length` is deprecated.  
 warnings.warn(

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding ( <a href="#">Embedding</a> )	?	0 (unbuilt)
spatial_dropout1d ( <a href="#">SpatialDropout1D</a> )	?	0
lstm ( <a href="#">LSTM</a> )	?	0 (unbuilt)
dense ( <a href="#">Dense</a> )	?	0 (unbuilt)

Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)

Starting model training for 5 epochs...

Epoch 1/5