Hate Speech Detection This project focuses on building a model combining LSTM and Hugging Face transformer models to detect hate speech from a curated text dataset. It not only classifies whether a statement is hateful but also predicts the intensity level of the hate speech. https://www.kaggle.com/datasets/waalbannyantudre/hate-speech-detection-curated-dataset/data

LSTM Model

```
pip install --upgrade kagglehub
```

```
Requirement already satisfied: kagglehub in /usr/local/lib/python3.11/dist-packages (0.3.12)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from kagglehub) (24.2)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from kagglehub) (6.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kagglehub) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kagglehub) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.4.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->kagglehub) (2025.4.26)
```

```
import kagglehub
waalbannyantudre_hate_speech_detection_curated_dataset_path = kagglehub.dataset_download('waalbannyantudre/hate-speech-detection-curated-

print('Data source import complete.')

import pandas as pd
import numpy as np
import tensorflow as tf
import random
from tensorflow.keras.layers import  Input,Dense,Embedding, LSTM,Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from nltk.corpus import stopwords
import re
import nltk
nltk.download('stopwords')
```

```
Data source import complete.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Enable dynamic memory growth for GPUs

```
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        for gpu_device in gpus:
            tf.config.experimental.set_memory_growth(gpu_device, True)
    except RuntimeError as e:
        print("GPU Memory Growth Error:", e)
```

Load dataset

```
dataset_path = "/kaggle/input/hate-speech-detection-curated-dataset/HateSpeechDatasetBalanced.csv"
df = pd.read_csv(dataset_path)
stop_words = set(stopwords.words('english'))
```

Clean text

```
def clean_text(text):
    text = re.sub(r'[^a-zA-Z]', ' ', text)
    text = text.lower()
    words = text.split()
    filtered_words = [word for word in words if word not in stop_words]
    return " ".join(filtered_words)

df['CleanedContent'] = df['Content'].apply(clean_text)
```

Tokenization and padding

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df['CleanedContent'])
sequences = tokenizer.texts_to_sequences(df['CleanedContent'])
max_sequence_length = max(len(seq) for seq in sequences)

X = pad_sequences(sequences, maxlen=max_sequence_length)
y = df['Label'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
original_test_texts = df['Content'].iloc[X_test.shape[0] * -1:].tolist()

# Set seeds
random.seed(42)
np.random.seed(42)
tf.random.set_seed(42)
```

Model definition

```
input_layer = Input(shape=(X_train.shape[1],))
embedding_layer = Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=20)(input_layer)
lstm_layer1 = LSTM(10, return_sequences=True)(embedding_layer)
lstm_layer2 = LSTM(10, return_sequences=True)(lstm_layer1)
flatten_layer = Flatten()(lstm_layer2)
output_layer = Dense(1, activation='sigmoid')(flatten_layer)
model = Model(inputs=input_layer, outputs=output_layer)
```

Compile and train

```
optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=5, batch_size=64, validation_data=(X_test, y_test))
```

```
Epoch 1/5
9077/9077 ━━━━━━━━━━━━━━━━━━━━ 2550s 280ms/step - accuracy: 0.8118 - loss: 0.4025 - val_accuracy: 0.8568 - val_loss: 0.3199
Epoch 2/5
9077/9077 ━━━━━━━━━━━━━━━━━━━━ 2569s 277ms/step - accuracy: 0.8775 - loss: 0.2808 - val_accuracy: 0.8650 - val_loss: 0.3169
Epoch 3/5
9077/9077 ━━━━━━━━━━━━━━━━━━━━ 2511s 273ms/step - accuracy: 0.8953 - loss: 0.2418 - val_accuracy: 0.8641 - val_loss: 0.3281
Epoch 4/5
9077/9077 ━━━━━━━━━━━━━━━━━━━━ 2466s 272ms/step - accuracy: 0.9039 - loss: 0.2230 - val_accuracy: 0.8649 - val_loss: 0.3352
Epoch 5/5
9077/9077 ━━━━━━━━━━━━━━━━━━━━ 2513s 277ms/step - accuracy: 0.9091 - loss: 0.2122 - val_accuracy: 0.8645 - val_loss: 0.3395
```

Evaluate on test data

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"\nTest Loss: {loss}")
print(f"Test Accuracy: {accuracy}")

# Predictions on test data
y_pred_probs = model.predict(X_test).squeeze()
y_pred_labels = (y_pred_probs > 0.5).astype(int)
print(f"\nAccuracy Score (sklearn): {accuracy_score(y_test, y_pred_labels)}")

# Evaluate on 20 random test sentences
print("\n--- Evaluation on 20 Test Sentences ---")
indices = np.random.choice(len(X_test), 20, replace=False)
for idx in indices:
    sentence = original_test_texts[idx]
    true_label = "HATE" if y_test[idx] == 1 else "NON-HATE"
    prob = y_pred_probs[idx]
    pred_label = "HATE" if prob > 0.5 else "NON-HATE"
    intensity = round(prob * 100, 2) if pred_label == "HATE" else round((1 - prob) * 100, 2)

    print(f"\nSentence: {sentence}")
    print(f"True Label: {true_label}")
    print(f"Predicted Label: {pred_label}")
    print(f"{'Hate' if pred_label == 'HATE' else 'Non-Hate'} Intensity: {intensity}%")
```

```
4539/4539 ━━━━━━━━━━━━━━━━━━━━ 223s 49ms/step - accuracy: 0.8639 - loss: 0.3405

Test Loss: 0.33951452374458313
Test Accuracy: 0.8644921183586121
4539/4539 ━━━━━━━━━━━━━━━━━━━━ 215s 47ms/step

Accuracy Score (sklearn): 0.8644920949705283

--- Evaluation on 20 Test Sentences ---

Sentence: stop vandalizing please stop your disruptive editing of removing rights to the unreliable and biased sources when you c
True Label: HATE
Predicted Label: HATE
Hate Intensity: 92.54000091552734%

Sentence: what a hypocritical award you have i discover that you have a b wizard in christianity on your user pageboy but you cal
True Label: HATE
Predicted Label: HATE
Hate Intensity: 83.51000213623047%

Sentence: an ii domain binds to a effort radiate dome
True Label: HATE
Predicted Label: HATE
Hate Intensity: 74.75%

Sentence: you lie in truth are dumb
True Label: HATE
Predicted Label: HATE
Hate Intensity: 77.62999725341797%

Sentence: his edits do suck which is why i reverted them over and over again not one edit contributed positively to the article w
True Label: HATE
Predicted Label: HATE
Hate Intensity: 99.19999694824219%

Sentence: i then suppose again i should not to be terribly surprised on wikipedia jews are hunted down and occasionally banned li
True Label: HATE
Predicted Label: HATE
Hate Intensity: 98.83999633789062%

Sentence: the notable occurrence at our borders is human trafficking and only did you know that there are many cases where daca w
True Label: NON-HATE
Predicted Label: NON-HATE
Non-Hate Intensity: 99.25%

Sentence: as you want and ignore some lump of data that shows islam to be and pile of crap it is your home
True Label: HATE
Predicted Label: HATE
Hate Intensity: 99.69000244140625%

Sentence: damn you arrogant smart little bitch you re very willing to toss the salad i got some peanut oil and jelly in my ass wa
True Label: HATE
Predicted Label: HATE
Hate Intensity: 87.83999633789062%

Sentence: death is not fit of ask that for what without good to the intolerant rapist warmonger thief murderer the pork be upon w
True Label: NON-HATE
```

predict label and hate percentage for custom sentences

```
# Function to predict label and hate percentage for custom sentences
def predict_custom_sentences(sentences):
    cleaned = [clean_text(sentence) for sentence in sentences]
    seqs = tokenizer.texts_to_sequences(cleaned)
    padded = pad_sequences(seqs, maxlen=max_sequence_length)
    probs = model.predict(padded).squeeze()

    for i, sentence in enumerate(sentences):
        prob = probs[i]
        label = "HATE" if prob > 0.5 else "NON-HATE"
        intensity = round(prob * 100, 2) if label == "HATE" else round((1 - prob) * 100, 2)
        print(f"\nSentence: {sentence}")
        print(f"Predicted Label: {label}")
        print(f"{'Hate' if label == 'HATE' else 'Non-Hate'} Intensity: {intensity}%")

# Example predictions on 4 custom sentences
custom_sentences = [
    "I absolutely despise those people!",
    "I hope you have a wonderful day!",
    "You're so stupid and annoying.",
    "This is a peaceful and loving community."
]

predict_custom_sentences(custom_sentences)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step

Sentence: I absolutely despise those people!
Predicted Label: HATE
Hate Intensity: 59.52000045776367%

Sentence: I hope you have a wonderful day!
Predicted Label: NON-HATE
Non-Hate Intensity: 95.81999969482422%

Sentence: You're so stupid and annoying.
Predicted Label: HATE
Hate Intensity: 88.68000030517578%

Sentence: This is a peaceful and loving community.
Predicted Label: NON-HATE
Non-Hate Intensity: 64.98999786376953%
```