Hate Speech Detection Using Natural Language Processing (NLP)

Objective

To build a Natural Language Processing model that automatically detects hate speech in text using a pre-trained Transformer (DistilBERT) model and evaluate its performance through accuracy, classification report, and confusion matrix.

Introduction

Natural Language Processing (NLP) enables computers to understand and process human language. Hate speech detection is essential for maintaining safety in online platforms. This experiment applies the DistilBERT model, a compact version of BERT, for text classification. The process includes text cleaning, tokenization, fine-tuning the model, evaluating performance, and deploying it using a Gradio interface for real-time prediction.

Methodology

Step 1: Imports and Dataset Path

Import essential libraries and specify the dataset file path.

```
DATA_PATH = "/content/drive/MyDrive/Colab Notebooks/MLL/MLL Datasets/HateSpeechDataset.csv"
```

Step 2: Load Dataset and Display Info

Read the dataset, display samples, and check label distribution.

```
Rows: 726119
```

	Content	Label
0	denial of normal the con be asked to comment o...	1
1	just by being able to tweet this insufferable ...	1
2	that is retarded you too cute to be single tha...	1
3	thought of a real badass mongol style declarat...	1
4	afro american basho	1

```
Label distribution:
Label
1    364525
0    361594
Name: count, dtype: int64
```

Step 3: Text Cleaning

Remove URLs, mentions, and punctuation, and convert text to lowercase.

```python
def clean_text_simple(s):
    if pd.isna(s):
        return ""
    s = str(s)
    s = re.sub(r"http\S+|www\.\S+", " ", s)
    s = re.sub(r"@\w+", " ", s)
    s = re.sub(r"[^A-Za-z0-9\s]", " ", s)
    s = re.sub(r"\s+", " ", s).strip()
    return s.lower()

df['text'] = df['Content'].apply(clean_text_simple)
df = df[['text','Label']].rename(columns={'Label':'label'})
df = df.drop_duplicates(subset=['text']).reset_index(drop=True)
df = df[df['text'].str.len() > 0].reset_index(drop=True)
print("After cleaning rows:", len(df))
display(df.head())
```

```
After cleaning rows: 699857
```

Step 4: Normalize Labels

Convert categorical labels into numeric format.

```python
unique_labels = sorted(df['label'].unique())
label2id = {lab:i for i,lab in enumerate(unique_labels)}
id2label = {i:lab for lab,i in label2id.items()}
df['label'] = df['label'].map(label2id).astype(int)
print("Label mapping:", label2id)
```

```
Label mapping: {np.int64(0): 0, np.int64(1): 1}
```

Step 5: Train-Test Split

Split the data into training and testing sets.

```python
from sklearn.model_selection import train_test_split

if len(unique_labels) > 1:
    train_df, test_df = train_test_split(df,test_size=0.20,random_state=42,stratify=df['label'])
else:
    train_df, test_df = train_test_split(df, test_size=0.20, random_state=42)

train_df = train_df.reset_index(drop=True)
test_df  = test_df.reset_index(drop=True)
print("Train shape:", train_df.shape, "Test shape:", test_df.shape)
```

```
Train shape: (559885, 2) Test shape: (139972, 2)
```

Step 6: Tokenization

Use DistilBERT tokenizer to convert text into numerical tokens.

```
MODEL_NAME = "distilbert-base-uncased"
MAX_LENGTH = 64

tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME, use_fast=True)

train_hf = Dataset.from_pandas(train_df[['text','label']])
test_hf  = Dataset.from_pandas(test_df[['text','label']])
```

Step 6.1: Limit Dataset Size

Select 50,000 samples for efficient yet effective training.

```
MAX_TRAIN = min(50000, len(train_tok))
MAX_TEST  = min(5000, len(test_tok))

train_tok = train_tok.select(range(MAX_TRAIN))
test_tok  = test_tok.select(range(MAX_TEST))
```

```
Train examples (subset): 50000
Test examples (subset): 5000
```

Step 7: Model Initialization

Initialize DistilBERT for sequence classification and define parameters.

```
from transformers import AutoModelForSequenceClassification, TrainingArguments, Trainer
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

num_labels = len(unique_labels)
model = AutoModelForSequenceClassification.from_pretrained(MODEL_NAME, num_labels=num_labels)
```

Step 8: Model Training

Train the model using the Hugging Face Trainer API.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_tok,
    eval_dataset=test_tok,
    compute_metrics=compute_metrics,
    tokenizer=tokenizer
)

trainer.train()
```

Step 9 & 10: Evaluation

Evaluate model on both training and testing data using standard metrics.

```
Train Accuracy: 0.98838
Train Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     24214
           1       0.99      0.98      0.99     25786

    accuracy                           0.99     50000
   macro avg       0.99      0.99      0.99     50000
weighted avg       0.99      0.99      0.99     50000

Train Confusion Matrix:
 [[24028   186]
 [  395 25391]]
```

```
Test Accuracy: 0.8528
Test Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.89      0.85      2408
           1       0.89      0.82      0.85      2592

    accuracy                           0.85      5000
   macro avg       0.85      0.85      0.85      5000
weighted avg       0.86      0.85      0.85      5000

Test Confusion Matrix:
 [[2134  274]
 [ 462 2130]]
```

Step 11: Inference for New Texts

Test the model with new unseen text samples.

```python
def predict_texts(texts):
    enc = tokenizer_inf(texts,truncation=True,padding=True,max_length=64,return_tensors='pt')
    if torch.cuda.is_available():
        enc = {k:v.cuda() for k,v in enc.items()}
    with torch.no_grad():
        out = model_inf(**enc)
        probs = F.softmax(out.logits, dim=-1).cpu().numpy()
        preds = probs.argmax(axis=-1)
    return preds, probs
```

Step 12: Sample Text & Predicted Label

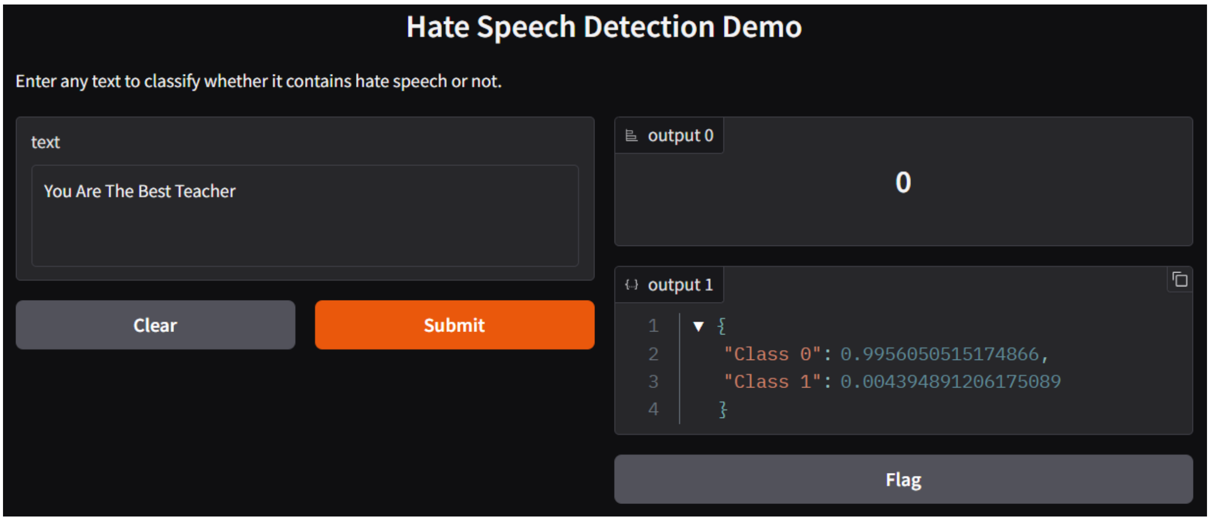
```
Text: I love this movie!
Predicted Label: 0, Probabilities: [0.98480105 0.01519898]

Text: He is a worst person
Predicted Label: 1, Probabilities: [0.0047696 0.9952304]

Text: I owe to you
Predicted Label: 0, Probabilities: [0.9979899  0.00201006]

Text: You are disgusting and should leave.
Predicted Label: 1, Probabilities: [0.03100662 0.9689934 ]
```

Step 13: Gradio Interface



Discussion

This experiment demonstrated the application of NLP techniques using the DistilBERT model for hate speech detection. The preprocessing steps ensured clean and structured input data. The model achieved high accuracy and strong generalization with 50,000 training samples. Evaluation metrics confirmed effective classification performance. The integration of a Gradio interface allowed easy interaction and real-time testing, showcasing how transformer-based NLP models can be practically deployed for automated text moderation and social media analysis.