Identify & Categorize Toxic Comments Online

BY:
ZAHID RAHMAN
KUSHAL REDDY SINGARAM
SAI SARAN RANGISETTI

Deep Learning - ISM 6561 | Professor Reza Ebrahimi | Spring 2025

Overview

1. Introduction
2. Executive Summary
3. Dataset & Data Prep
4. Model Overviews
5. Comparison Summary
6. Live Demo

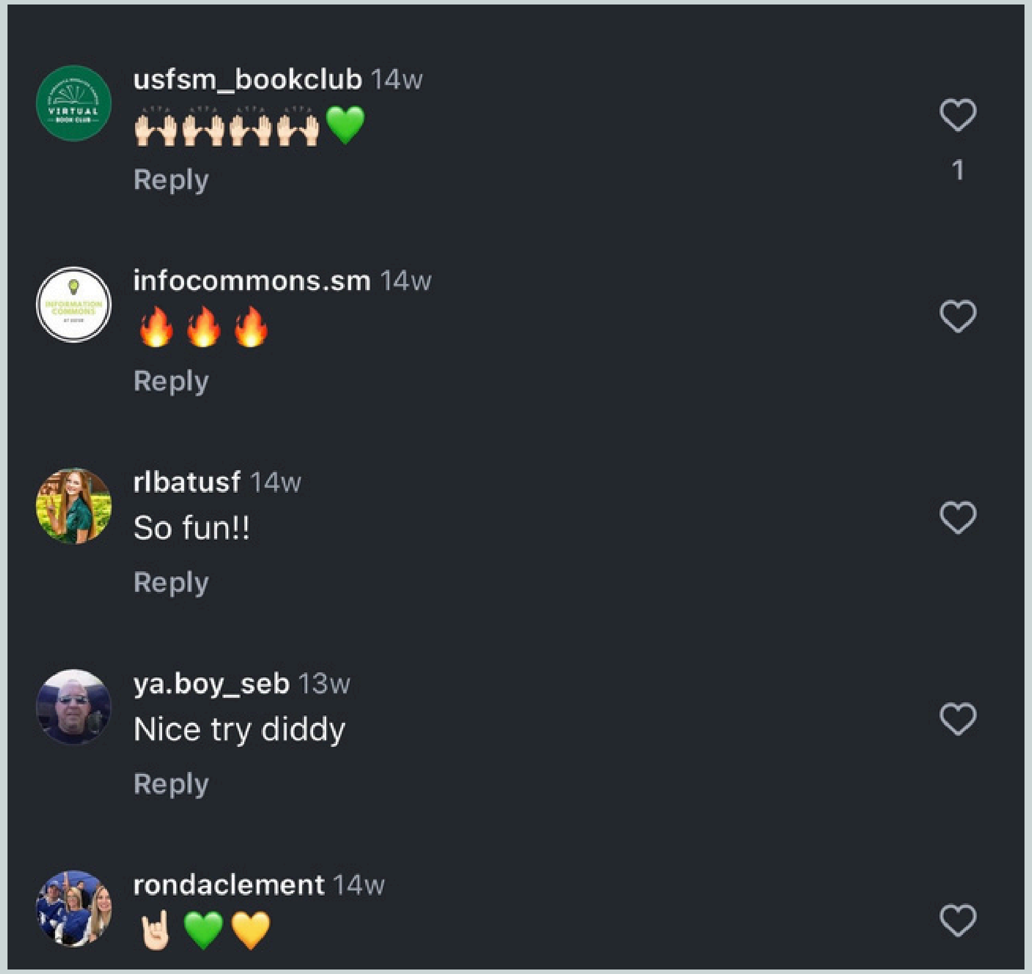
Introduction

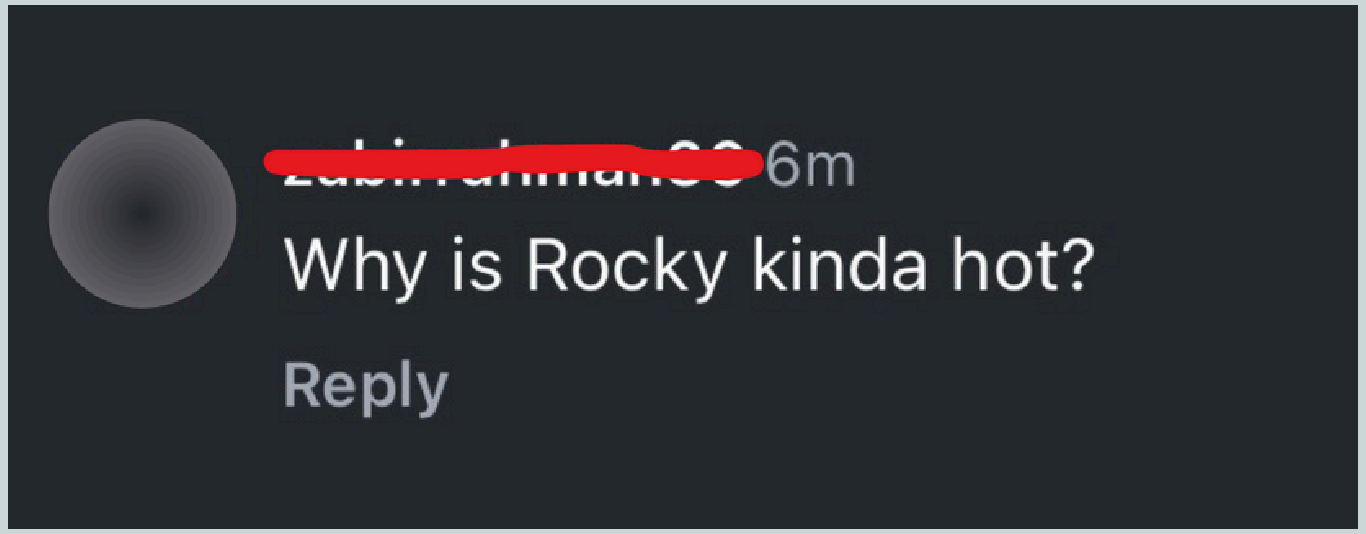
- Social media brings the world onto one platform.
- Everyone has different opinions yet deserves the freedom to share them.

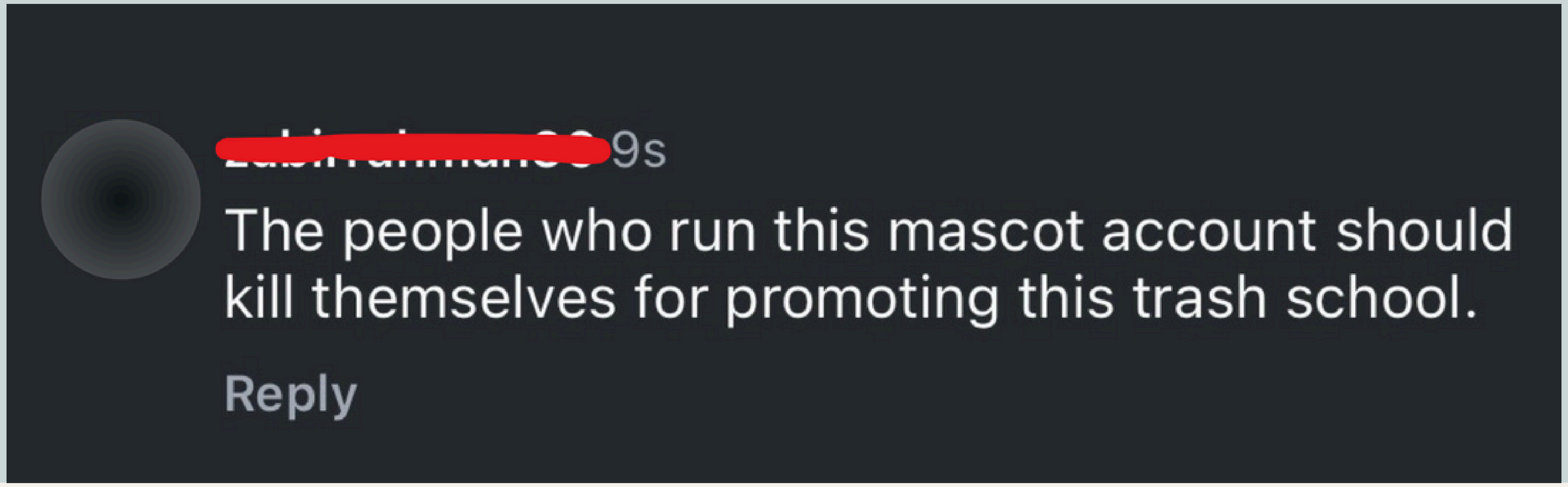
- However, many spout toxicity or hate speech -- but they wouldn't say it in person
- Being behind a screen emboldens us or helps us feel anonymous.
- Toxic comments often lead to more toxic replies, creating a negative cycle

- **<u>It is critical to detect and manage harmful content online.</u>**

Introduction - Business Application







Toxic Comment Multi-Label Classification Objective

Build a multi-label text classification system to identify toxic comments across six categories: toxic, severe toxic, obscene, threat, insult, and identity hate using DistilBERT and DeBERTa transformer models.

Ultimately helping platforms automatically flag harmful content, improve user experience and protect brand reputation

Dataset & Data Prep

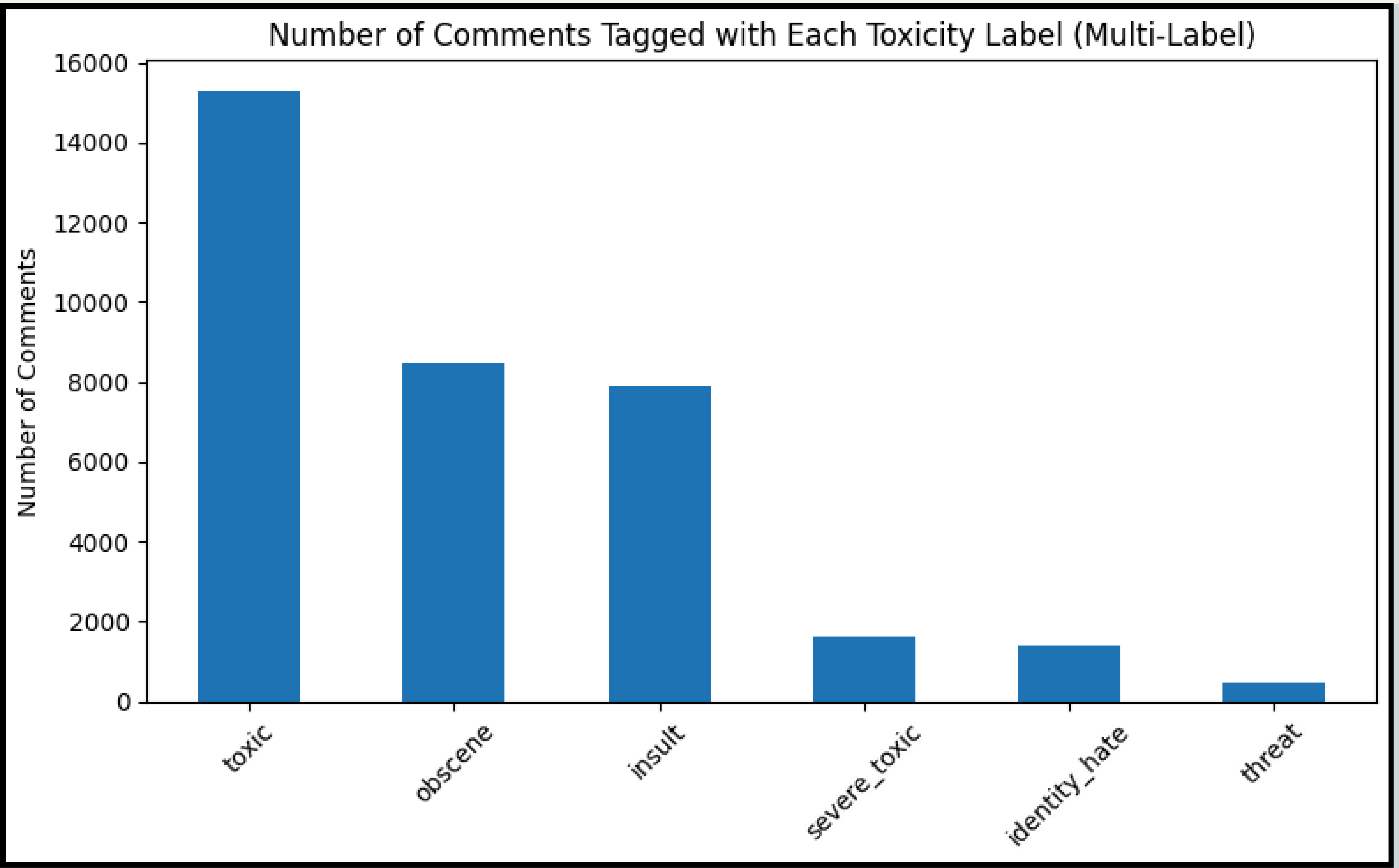
- 160,000 rows
    - Comment_Text
    - Output Categories
        - Multi-label classification: Each comment can belong to any combination of the six toxicity categories (including none)

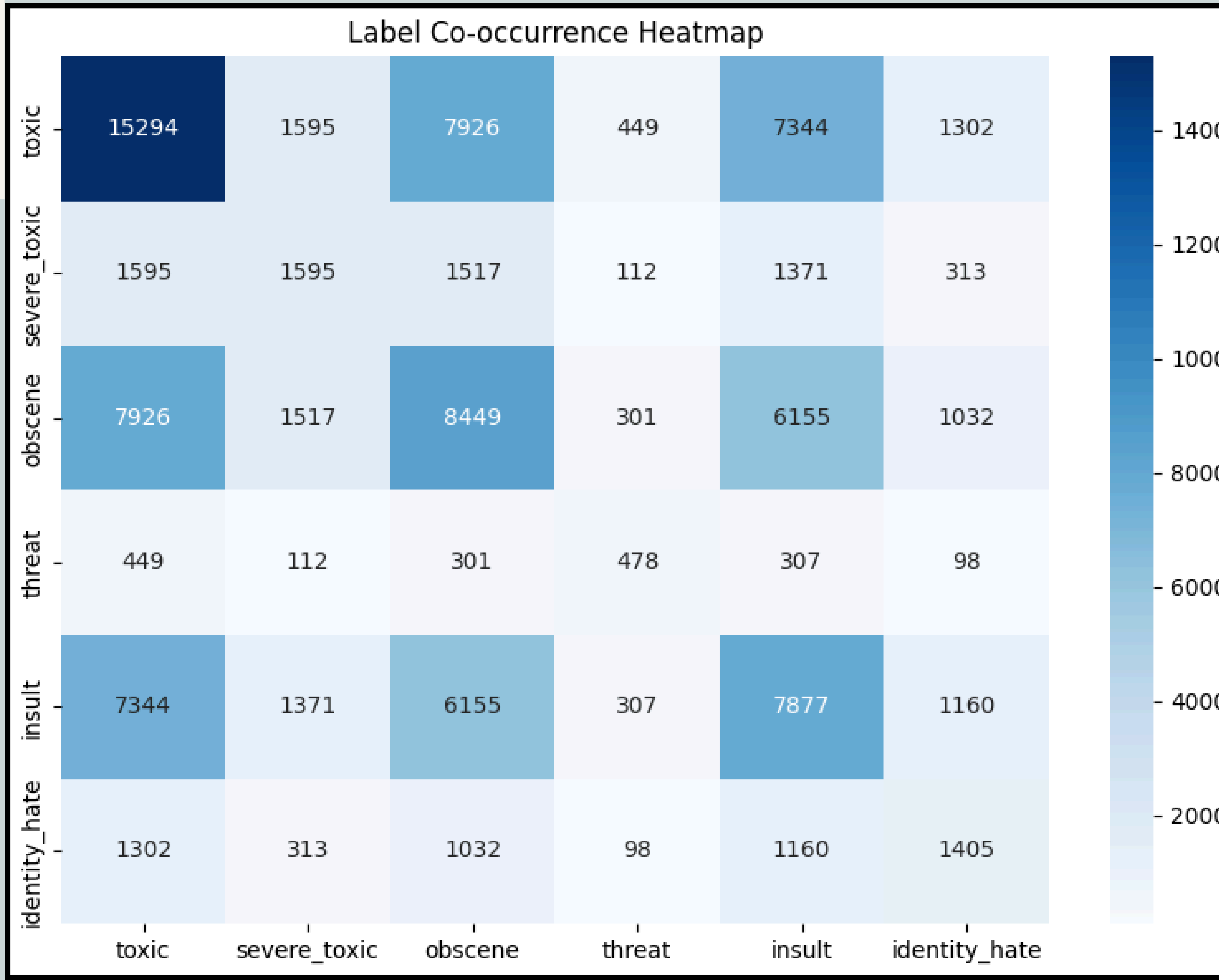
comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
WOULDN'T BE THE FIRST TIME BITCH. FUCK YOU I'LL FIND OUT WHERE YOU LIVE, SODOMIZE YOUR WIFE AND THEN BURN YOUR HOUSE DOWN. FUCK YOU YOU FUCKING QUEER.	1	1	1	1	1	1
SHUT UP, YOU FAT POOP, OR I WILL KICK YOUR ASS!!!	1	1	1	1	1	1
You're a stupid cunt Fuck you dumb arse, your mum has a hairy cunt and I hope and pray that you die, no, fuck that, I wish you would die, if I had three wishes, one of them would be to have you dead,						

Dataset & Data Prep

Most toxic behavior online is concentrated in general categories like 'toxic', 'obscene', and 'insult', which also tend to appear together.

This suggests that moderation systems should prioritize detecting these frequent and overlapping forms of toxicity to have the greatest real-world impact.





Dataset & Prep - Tokenization & Encoding

- Split into 80% training, 10% validation, and 10% test
- Isolated target toxicity categories
- Tokenized comments using Hugging Face’s DistilBERT and DeBERTa tokenizers
- Set max_length=128 and converted outputs to PyTorch tensors
- Prepared raw text into numerical format required for transformer models

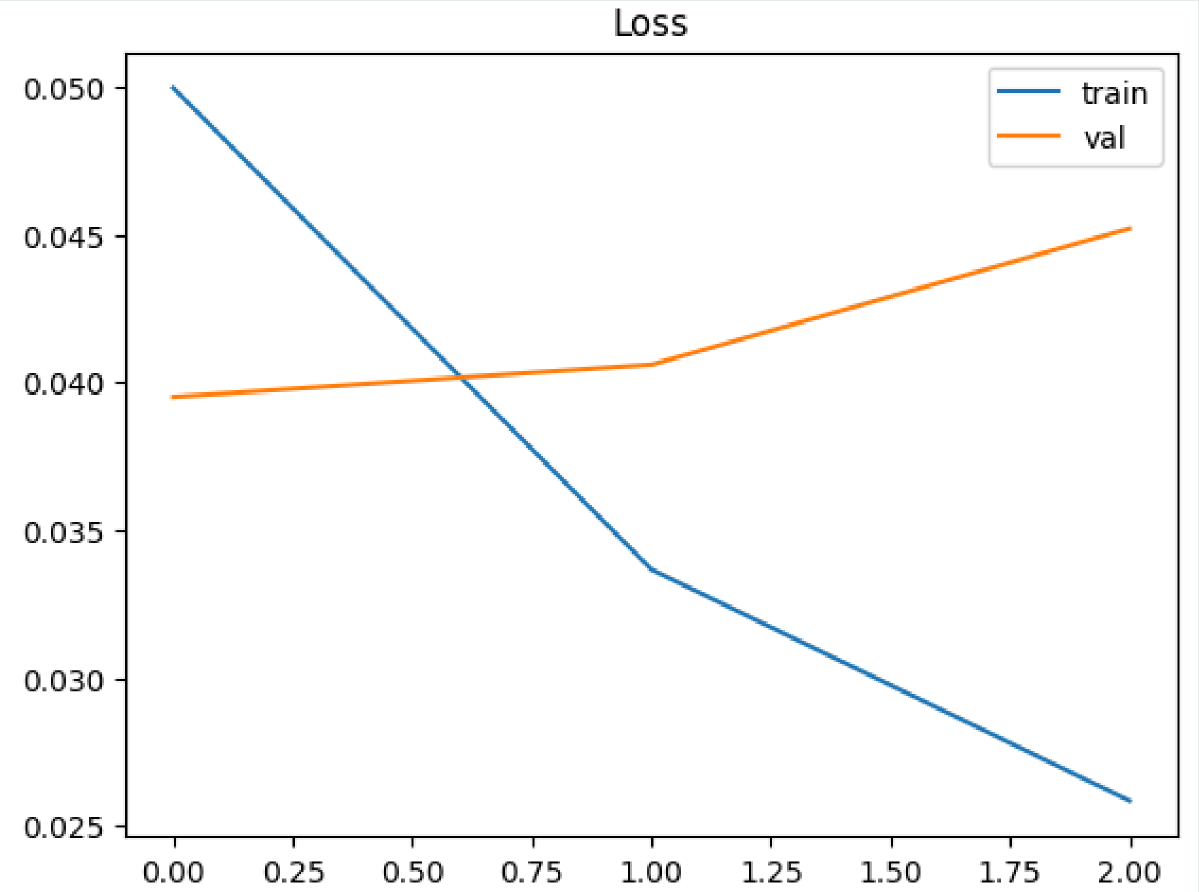
```
[ ]
# ----------------------------------------------------------------------
# 2. Model Building, Compilation, and Training with DistilBERT
# ----------------------------------------------------------------------
# NOTE: Tokenization was performed offline in a separate notebook.
# The tokenized data (with keys "input_ids", "attention_masks", and "labels")
# was saved to 'tokenized_data.pkl' and then loaded and split into train/validation/test datasets.
#
# We now build and fine-tune a pre-trained, distilled transformer (DistilBERT)
# configured for multi-label classification on our toxicity dataset.

# load pre-tokenized arrays
with open('tokenized_data.pkl','rb') as f:
    data = pickle.load(f)
input_ids, attention_masks, labels = (data[k] for k in ['input_ids','attention_masks','labels'])

# train/val/test splits
n = len(input_ids)
i_train = int(0.7*n); i_val = i_train + int(0.2*n)
splits = {
    'train': (input_ids[:i_train], attention_masks[:i_train], labels[:i_train]),
    'val':   (input_ids[i_train:i_val], attention_masks[i_train:i_val], labels[i_train:i_val]),
    'test':  (input_ids[i_val:], attention_masks[i_val:], labels[i_val:])
}

# dataset builder
def make_ds(ids, masks, labs, batch=16, buf=10000):
    ds = tf.data.Dataset.from_tensor_slices(
        ({"input_ids":ids,"attention_mask":masks}, labs)
    )
    return ds.shuffle(buf, seed=42).batch(batch).cache().prefetch(tf.data.AUTOTUNE)

train_ds = make_ds(*splits['train'])
val_ds   = make_ds(*splits['val'])
test_ds  = make_ds(*splits['test'])
```

Models

- DistilBERT + Binary Cross Entropy
- DeBERTa + Focal Loss + Label Smoothing
- DeBERTa + FGSM Adversarial Training

Model I: DistilBERT + Binary Cross Entropy Baseline


```
# Took 13m 15s to run using PAID compute (A100)
# instantiate & compile
model = TFDistilBertForSequenceClassification.from_pretrained(
    'distilbert-base-uncased', num_labels=6, problem_type='multi_label_classification'
)
model.compile(
    optimizer=tf.keras.optimizers.Adam(2e-5),
    loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
    metrics=['accuracy']
)
model.summary()
```



Label	Precision	Recall	F1-Score	Support
toxic	0.83	0.82	0.83	1543
severe_toxic	0.37	0.69	0.48	150
obscene	0.78	0.86	0.82	864
threat	0.50	0.52	0.51	50
insult	0.75	0.76	0.75	817
identity_hate	0.53	0.65	0.59	144
micro avg	0.75	0.80	0.77	3568
macro avg	0.63	0.72	0.66	3568
weighted avg	0.77	0.80	0.78	3568
samples avg	0.07	0.08	0.07	3568

Model II: DeBERTa + Focal Loss + Label Smoothing

```
# Load DeBERTa model for sequence classification (Model II)
model_deberta_focal = AutoModelForSequenceClassification.from_pretrained(
    'microsoft/deberta-v3-base',
    num_labels=NUM_LABELS,
    problem_type="multi_label_classification"
)
model_deberta_focal = model_deberta_focal.cuda()

# Define Focal Loss with label smoothing
class FocalLoss(nn.Module):
    def __init__(self, gamma=2.0, smoothing=0.1):
        super(FocalLoss, self).__init__()
        self.gamma = gamma
        self.smoothing = smoothing
    def forward(self, logits, targets):
        # Apply label smoothing: y_ls = y*(1-alpha) + alpha/2 for each label
        if self.smoothing > 0:
            targets = targets * (1 - self.smoothing) + 0.5 * self.smoothing
        # Compute binary cross entropy with logits for each label (no reduction)
        bce = nn.functional.binary_cross_entropy_with_logits(logits, targets, reduction='none')
        # Convert to probability space for focal scaling factor pt
        # p_t = exp(-bce) as given by the focal loss formula
        pt = torch.exp(-bce)
        # Compute focal loss scaling factor
        focal_factor = (1 - pt) ** self.gamma
        # Apply focal factor to BCE loss
        loss = focal_factor * bce
        # Average loss over all samples and labels
        return loss.mean()

criterion_focal = FocalLoss(gamma=2.0, smoothing=0.1)
optim_deberta_focal = torch.optim.AdamW(model_deberta_focal.parameters(), lr=2e-5)
```

- Used **microsoft/deberta-v3-base** with custom Focal Loss (gamma=2, smoothing=0.1).

- Trained for 2 epochs.

Results:

- F1 Scores: Obscene (0.84), Toxic (0.81), Identity Hate (0.63)
- Weighted Avg F1: 0.80

Why: DeBERTa provides deeper contextual understanding; Focal Loss combats class imbalance.

Label	Precision	Recall	F1-Score	Support
toxic	0.84	0.83	0.83	1520
severe_toxic	0.38	0.78	0.51	162
obscene	0.84	0.81	0.83	856
threat	0.39	0.57	0.46	37
insult	0.78	0.77	0.78	808
identity_hate	0.54	0.54	0.54	138

Model III: DeBERTa + FGSM Adversarial Training


```
num_epochs = 2
model_deberta_adv.train()
for epoch in range(num_epochs):
    total_loss = 0.0
    for batch in train_loader_deberta:
        input_ids = batch['input_ids'].cuda()
        attention_mask = batch['attention_mask'].cuda()
        labels = batch['labels'].cuda()
        # Step 1: Forward pass on clean inputs
        # We will manually get embeddings to apply FGSM
        # Get embedding output for input ids
        embeddings = model_deberta_adv.base_model.embeddings.word_embeddings(input_ids)
        embeddings.retain_grad()  # retain grad on embeddings for FGSM
        outputs_clean = model_deberta_adv(inputs_embeds=embeddings, attention_mask=attention_mask)
        logits_clean = outputs_clean.logits
        loss_clean = nn.functional.binary_cross_entropy_with_logits(logits_clean, labels)
        # Step 2: Backpropagate to get gradients w.rt embeddings
        optim_deberta_adv.zero_grad()
        loss_clean.backward(retain_graph=True)  # compute grad, keep graph for second pass
        # Step 3: FGSM perturbation on embeddings
        grad = embeddings.grad.detach()                 # gradient of loss wrt embeddings
        perturbation = epsilon * torch.sign(grad)       # compute perturbation
        embeddings_adv = embeddings + perturbation      # adversarial embeddings
        # Step 4: Forward pass with adversarial embeddings
        outputs_adv = model_deberta_adv(inputs_embeds=embeddings_adv.detach(), attention_mask=attention_mask)
        logits_adv = outputs_adv.logits
        loss_adv = nn.functional.binary_cross_entropy_with_logits(logits_adv, labels)
        # Step 5: Combine losses (we average them to balance importance)
        total_batch_loss = 0.5 * loss_clean + 0.5 * loss_adv
        # Step 6: Backpropagate combined loss and update weights
        optim_deberta_adv.zero_grad()  # clear gradients (note: also cleared embeddings.grad)
        total_batch_loss.backward()
        optim_deberta_adv.step()
        total_loss += total_batch_loss.item()
    avg_loss = total_loss / len(train_loader_deberta)
    print(f"Epoch {epoch+1} - Model III Adv Training Loss: {avg_loss:.4f}")
```

```
Some weights of DebertaV2ForSequenceClassification were not initialized from the model checkpoint at microsoft/deberta-v3-base and are newly initial:
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1 - Model III Adv Training Loss: 0.0620
Epoch 2 - Model III Adv Training Loss: 0.0518
```

- Adversarial Fine-Tuning with DeBERTaV3 (FGSM-based)
- Added adversarial perturbation (epsilon = 0.1) using FGSM on word embeddings.
- Combined clean and perturbed loss equally during training.

Results:

- F1 Scores: Toxic (0.84), Obscene (0.83), Insult (0.78)
- Weighted Avg F1: 0.78

Classification report for Model III


```
test_logits = []
test_true = []
with torch.no_grad():
    for batch in test_loader_deberta:
        input_ids = batch['input_ids'].cuda()
        attention_mask = batch['attention_mask'].cuda()
        labels = batch['labels'].numpy()
        outputs = model_deberta_adv(input_ids=input_ids, attention_mask=attention_mask)
        logits = outputs.logits.cpu().numpy()
        test_logits.append(logits)
        test_true.append(labels)
test_logits = np.concatenate(test_logits, axis=0)
test_true = np.concatenate(test_true, axis=0)
test_probs = 1 / (1 + np.exp(-test_logits))
test_preds = (test_probs >= np.array(best_thresholds_model3)).astype(int)
print("\nClassification Report for Model III (DeBERTa + Adv Training):\n")
print(classification_report(test_true, test_preds, target_names=LABEL_COLS, zero_division=0))
```

```
Classification Report for Model III (DeBERTa + Adv Training):

               precision    recall  f1-score   support

        toxic       0.84      0.84      0.84      1520
 severe_toxic       0.47      0.63      0.54       162
      obscene       0.83      0.83      0.83       856
       threat       0.44      0.19      0.26        37
       insult       0.74      0.83      0.78       808
identity_hate       0.46      0.54      0.49       138

    micro avg       0.77      0.81      0.79      3521
    macro avg       0.63      0.64      0.62      3521
 weighted avg       0.78      0.81      0.79      3521
  samples avg       0.07      0.07      0.07      3521
```

- evaluation results of Model III (DeBERTa + Adversarial Training + Threshold Tuning) on the test set using the classification report. Key observations include:
- Highest F1-score for the "Toxic" label: 0.84
- Good performance on "Obscene" (F1 = 0.83) and "Insult" (F1 = 0.78)
- Moderate performance on "Severe Toxic" and "Identity Hate"
- Lower F1-score on "Threat" (F1 = 0.26)

Model III on validation set for threshold tuning

- Used sigmoid + threshold sweep on validation set to find optimal cutoffs.
- Applied tuned thresholds to test set.

Why: Needed for multi-label outputs to optimize F1-score per label.

```
[ ]  # Evaluate Model III on validation set for threshold tuning
     model_deberta_adv.eval()
     val_logits = []
     val_true = []
     with torch.no_grad():
         for batch in val_loader_deberta:
             input_ids = batch['input_ids'].cuda()
             attention_mask = batch['attention_mask'].cuda()
             labels = batch['labels'].numpy()
             outputs = model_deberta_adv(input_ids=input_ids, attention_mask=attention_mask)
             logits = outputs.logits.cpu().numpy()
             val_logits.append(logits)
             val_true.append(labels)
     val_logits = np.concatenate(val_logits, axis=0)
     val_true = np.concatenate(val_true, axis=0)
     val_probs = 1 / (1 + np.exp(-val_logits))

     best_thresholds_model3 = []
     for i in range(NUM_LABELS):
         y_true = val_true[:, i]
         y_prob = val_probs[:, i]
         best_thr = 0.5
         best_f1 = 0.0
         for thr in np.linspace(0, 1, 101):
             y_pred = (y_prob >= thr).astype(int)
             score = f1_score(y_true, y_pred, zero_division=0)
             if score > best_f1:
                 best_f1 = score
                 best_thr = thr
         best_thresholds_model3.append(best_thr)
         print(f"Label {LABEL_COLS[i]:<12}: best threshold = {best_thr:.2f}, F1 = {best_f1:.3f}")
```

Label	Best Threshold	F1 Score
toxic	0.59	0.837
severe_toxic	0.55	0.513
obscene	0.50	0.847
threat	0.17	0.282
insult	0.56	0.755
identity_hate	0.46	0.569

Gradio Interface for Comparison

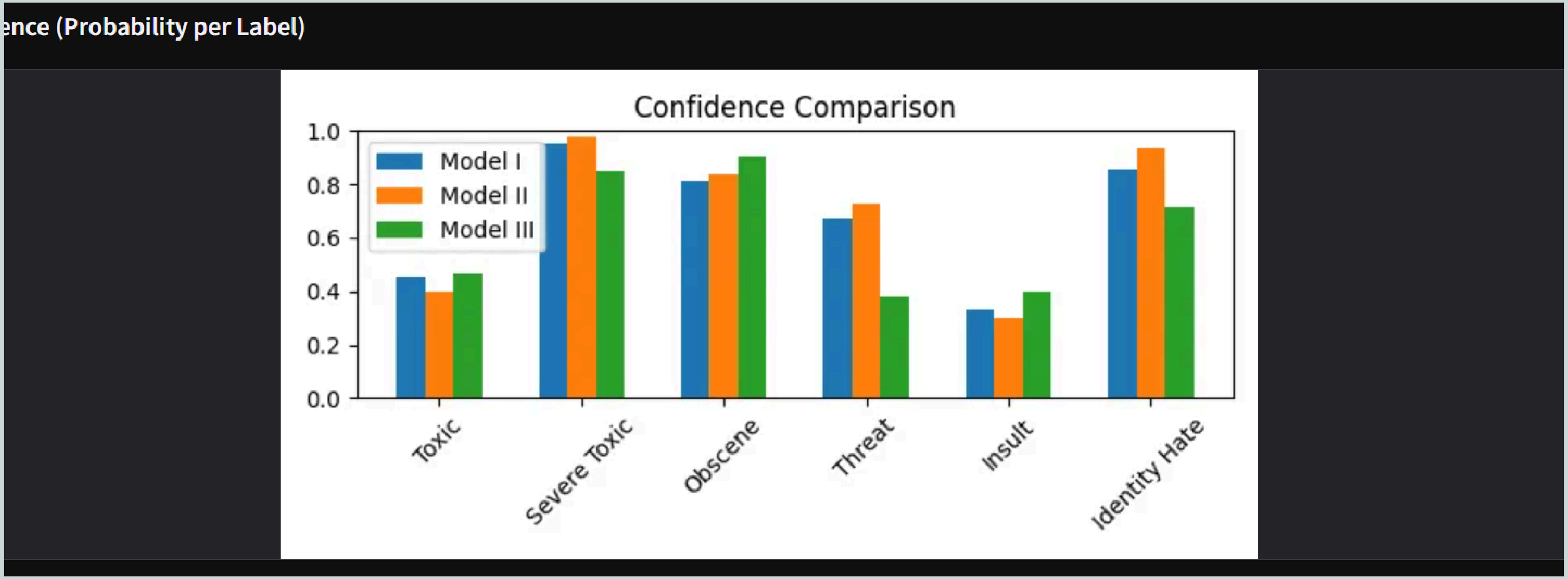
- **Enter a comment and get a predictions across all 3 models.**
- **A confidence bar chart and token-level attention highlights explainability, improving trust.**

Click for Gradio Demo

Performance Summary

Label	DistilBERT	DeBERTa + Focal	DeBERTa + Adv
Toxic	0.83	0.81	0.84
Obscene	0.85	0.84	0.83
Insult	0.76	0.74	0.78
Identity Hate	0.56	0.63	0.54
Macro F1	0.71	0.73	0.62
Weighted F1	0.79	0.80	0.78

Confidence Comparision



Final Takeaways

- DistilBERT is fast and efficient but struggles with less common instances.
- DeBERTa w/ Focal Loss enhances overall performance, while adversarial training ensures robustness without sacrificing accuracy.
- Combining attention and intuitive Gradio interface empowers users with transparency.

Today, we talk online more than in person—if we don't control toxicity, the internet won't be safe for anyone

Members	Contribution
Zahid	Model I + Attention Explainability + Gradio
Kushal	Model II: DeBERTa + Focal Loss
Sai	Model III: DeBERTa + Adversarial TrainingModel +Threshold Tuning

THANK YOU!