Bias, Censorship, and Accuracy: AI Models for Online Moderation

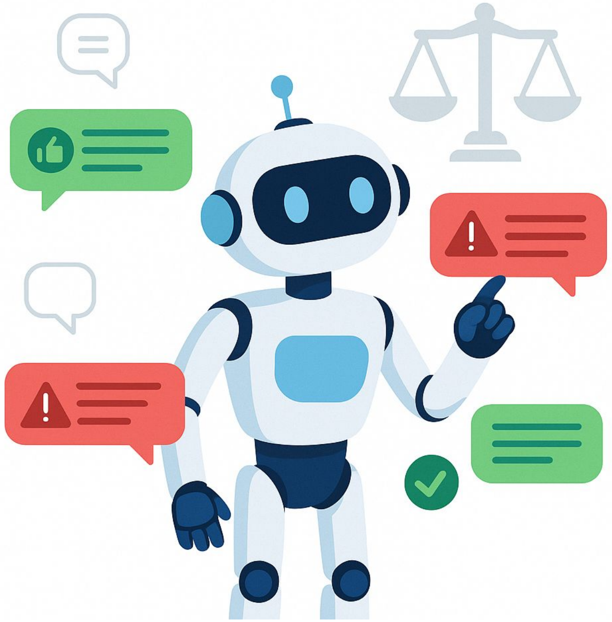
Josef Karpinski (jjk21004)
Alexander Manos (agm21019)
Samuel Cultrera (sac19019)
Malcolm Ferguson (mrf20004)

Project Overview

- Compared two AI models in their ability to classify online text comments as toxic or non-toxic
- Trained models using the Jigsaw Toxic Comment Classification Dataset

We discussed...

- Ethical concerns related to automatic content moderation and free speech
- Practical use and real world implications of these algorithms



Exploring the Dataset

Jigsaw Toxic Comment Classification Dataset

Purpose

- Widely used to develop machine learning models that aim to classify comments as toxic or non-toxic

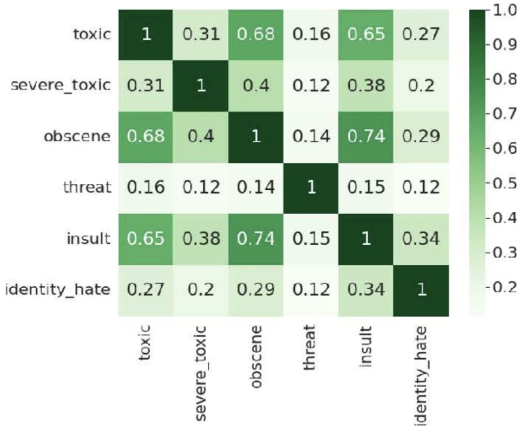
Why We Chose This Dataset

- Well-labeled data, ideal for supervised learning
- Real world use case

Specifications

Label	Positive	Negative	Proportion
toxic	15294	144277	10.6%
severe toxic	1595	157976	1.0%
obscene	8449	151122	5.6%
threat	478	159093	0.3%
insult	7877	151694	5.2%
identity hate	1405	158166	0.9%
total	15294	144277	10.6%

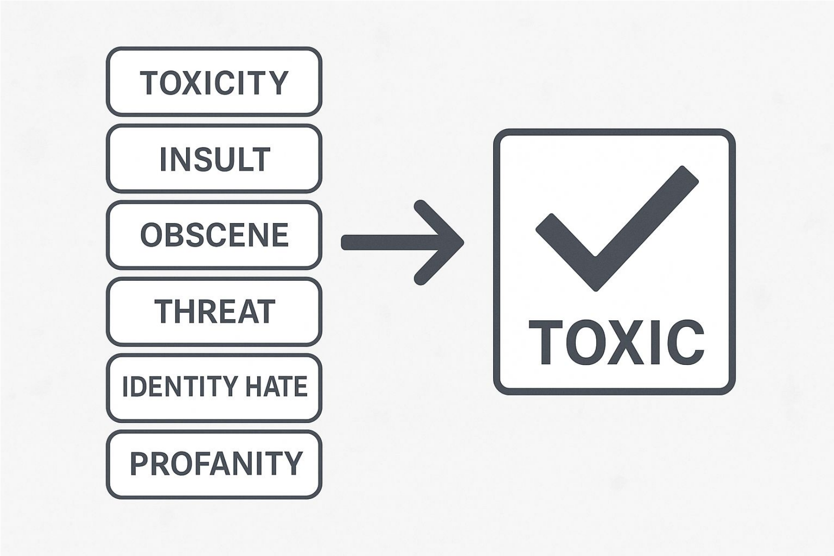
- <u>Number of Samples:</u> ~160,000
- <u>Data-Type:</u> English text comments from wikipedia talk pages
- <u>Format:</u> Multi-label classification
    - 6 Labels - toxic, severe toxic, obscene, threat, insult, identity hate



Methodology

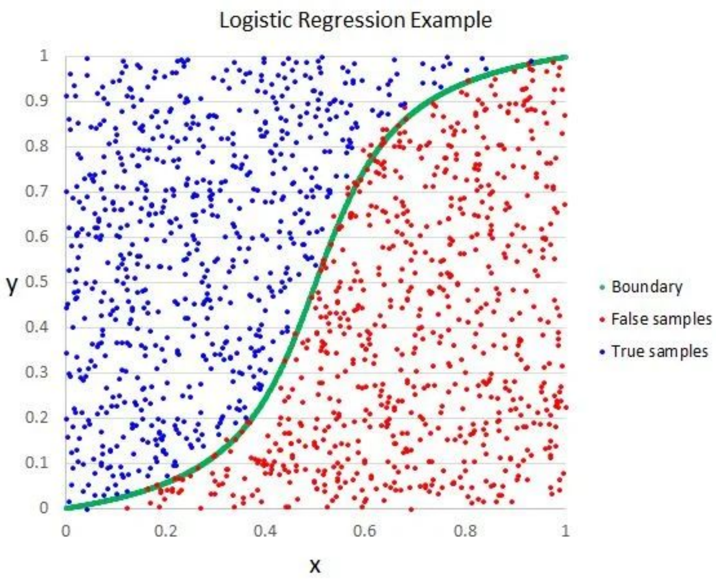
Combining Labels

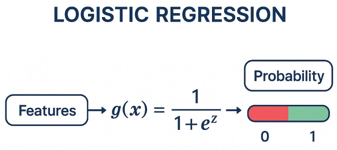
- For simplicity, we will combine our 6 labels in one binary label
- We will consider a text sample toxic if just 1 of the 6 labels is positive
- This turns the problem into binary classification, instead of multi classification



Logistic Regression

- Splitting the dataset
  - We split the dataset using an 80-20 split, meaning 80% training and 20% testing
- Vectorization of text data
  - Since we are dealing with text data, it is important to vectorize the data before we input it into our model
  - Use TfidfVectorizer from sklearn.feature_extraction.text
- Defining the model and prediction
  - Use LogisticRegression from sklearn.linear_model





Logistic Regression (Code)

- Splitting the dataset

```
X_train, X_test, y_train, y_test = train_test_split(
    df['comment_text'], # input
    df['toxic_any'], # output
    test_size=0.2, # 20% test, 80% train
    random_state=42
)
```

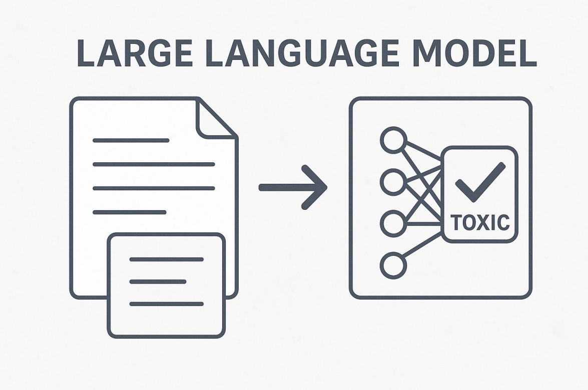
- Vectorization of text data

```
vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

- Defining the model and prediction

```
log_reg_model = LogisticRegression(max_iter=1000)
log_reg_model.fit(X_train_vec, y_train)
y_pred_log_reg = log_reg_model.predict(X_test_vec)
```

Large Language Model



- Unbiased Toxic RoBERTa Multiclassifier
  - Hugging Face: https://huggingface.co/unitary/unbiased-toxic-roberta
  - Trained on our dataset and a few others
- Sampling the dataset
  - Classification with an LLM is quite slow, so we drew a subset of 100 samples to do testing
- Testing and Classification
  - Run the LLM on each sample to get score for all of the LLM labels
  - If any of our desired scores are above a threshold, we classify the text as toxic
  - We must define our own classification function

LLM (Code 1)

- Downloading the classifier

```
llm_classifier = pipeline("text-classification", model="unitary/unbiased-toxic-roberta")
```

- Dataset Sampling

```
df_llm = df.sample(1000, random_state=42)

X_llm = df_llm['comment_text']
y_llm = df_llm['toxic_any']
```

LLM Labels:

- toxicity
- severe_toxicity
- obscene
- identity_attack
- insult
- threat
- sexual_explicit

LLM (Code 2)

- Custom function for classification

```python
def classify_llm(text, threshold=0.5):
    try:
        predictions = llm_classifier(text, top_k=None)
        # see if any of our labels are above the threshold
        for pred in predictions:
            if pred['label'] in TOXIC_LABELS:
                if pred['score'] > threshold:
                    return 1
        # if here, none of the labels were sufficient
        return 0
    except:
        return 0
```

```python
TOXIC_LABELS = {
    "toxicity",
    "severe_toxicity",
    "obscene",
    "threat",
    "insult",
    "identity_attack",
    "sexual_explicit"
}
```

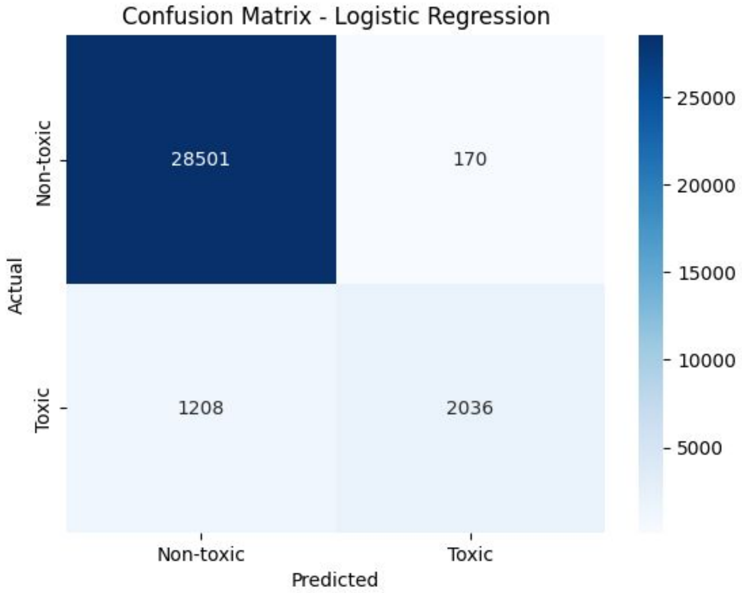
- Run LLM on test data

```python
y_pred_llm = X_llm.apply(classify_llm, threshold=0.5)
```

Results and Analysis

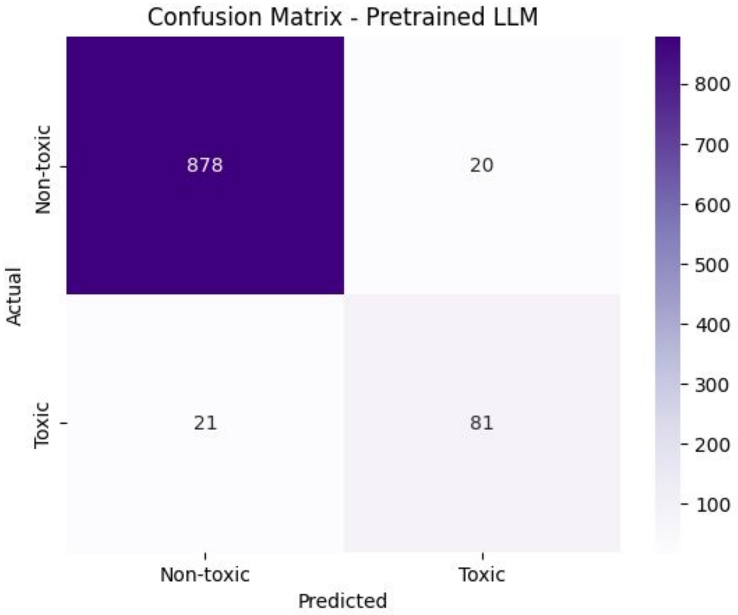
Results - Our Model

- Results from running on test data:
  - Precision: 0.9229
  - Recall: 0.6276
  - F1-score: 0.7473
- Overall accuracy = 0.9568



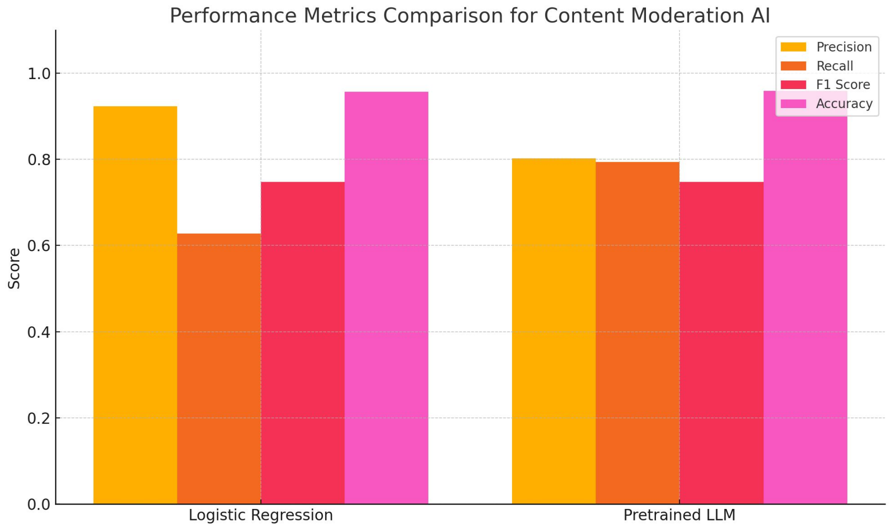
Results - LLM

- Results from running on test data:
  - Precision: 0.8020
  - Recall: 0.7941
  - F1-score: 0.7473
- Overall accuracy = 0.9590



Results - Comparison

- Overall accuracy very similar
- LLM has a 17% higher recall
    - More false positives, but less false negatives
    - How should this be balanced?
- Must take data balance into account when analyzing accuracy



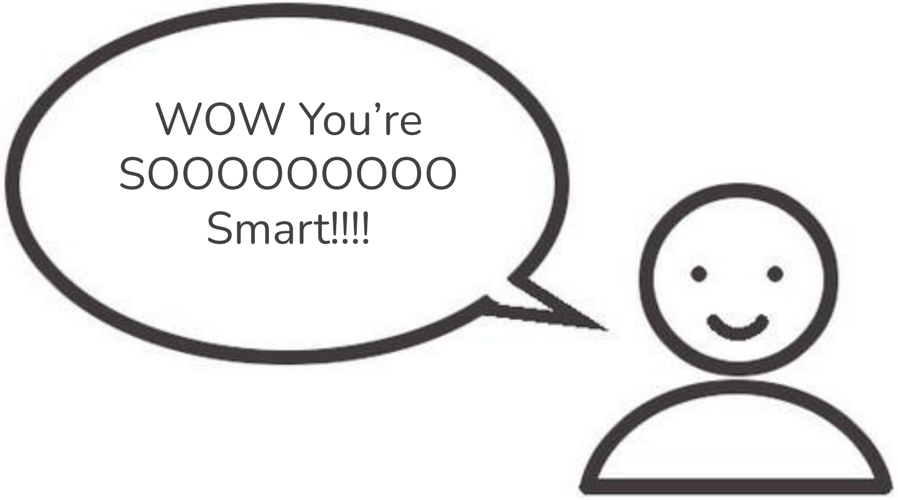
Ethical Discussion

Accuracy

AI often lacks nuance and cannot process context such as sarcasm

Risk of false negatives or false positives

It is *Very* Difficult for Artificial Intelligence to Fully Interpret Messages



Privacy Vs Protection

Since AI Cannot Reach 100% Accuracy, a Choice Must Be Made

Under Flagging

- Preserves freedom of expression
- Reduces false positives
- Allows harmful content to persist
- Erodes platform trust

Over Flagging

- Protects users
- May restrict free speech
- Suppresses legitimate expression
- Frustrates users

Potential Solution

Questions?

References

Jigsaw Toxic Comment Classification Dataset:
https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data

Unbiased Toxic RoBERTa Pre-trained LLM:
https://huggingface.co/unitary/unbiased-toxic-roberta

GitHub Repository
https://github.com/josef-karpinski/content-moderation-cse3000