# Detecting Toxic Content from Social Media using LLMs

Naman Chhibbar, Meka Nani

Indian Institute of Technology Hyderabad

27 September 2024

*Warning: This presentation contains sensitive content*

# Introduction

- Toxicity in natural language refers to expressing hate or stereotypes towards an individual or a group of people using language.
- In general, toxic content can be categorized into:
  - **Derogation:** Insulting or demeaning a group of individuals.
  - **Animosity:** Subtly abusing a group (without using explicit language).
  - **Threatening:** Expression of intent to harm, supporting harmful act, or encouraging inflicting harm on an individual or a group.
  - **Supporting toxicity:** Encouraging, justifying, or glorifying hateful acts.
  - **Dehumanizing:** Treating an individual or a group as "less human" (for example, insects, animals, etc.).

# Problem Statement

- Given some social media content (in the form of natural language), we need to predict whether it is toxic (i.e. a binary classification problem).
- Formally speaking, given a natural language input $x = (w_1, w_2, \ldots, w_n)$, we need to predict the probability that $x$ is toxic, which acts as a toxicity measure for $x$.

# Datasets

- ToxicGen
  - Created by prompting an LLM to produce toxic and benign texts.
  - Texts do not contain explicit language and target minority groups.
  - Example: Muslims deserve to die.
- SBIC (Social Bias Inference Corpus):
  - Texts contain stereotypes and offensiveness towards an individual or a group of people.
  - Example: Women candidates are less qualified.
- DHate
  - Details about this dataset are not revealed.
  - From manual analysis, this dataset seems to be comprised of hate of minorities.
  - Example: I always feel unsafe when surrounded by Arabs.

# Challenges

- A major challenge in detecting toxicity is that toxicity is relative.
- Some may think that some content is toxic, while others may not.
- This happens due to different perceptions of people.
- In addition, the datasets being used are also biased.
- For example:
  - "I'm going to kill Amy" is labelled as not toxic in the DHate dataset, whereas it falls in the "Threatening" category.
  - "A dude with money can only impress a broke bitch" is labelled as not toxic in the SBIC dataset, whereas many may think it is (due to vulgarity).

# Efficient Toxic Content Detection by Bootstrapping and Distilling Large Language Models

- Several deep learning techniques have been developed to automate toxic content detection
- **Supervised learning**: Predict a manually provided target output
  - Performance of the system grows with the size of labeled data
  - Data and annotations are rare, costly, or time-consuming to collect
  - Overfit to the training task
  - Lack the properties for knowledge transfer and generalization
- **Large Language Models**: superior zero-shot and few-shot context learning performance and transferability.
  - Designing novel prompting approaches to enhance the performance

# Efficient Toxic Content Detection by Bootstrapping and Distilling Large Language Models

- Performance relies heavily on the quality of prompts
- Deploying LLMs for toxic content detection can incur both high run-time costs and high latency
- **Existing works**-Bootstrapping and Distilling LLM's
  - Decision-Tree-of-Thought
  - Fine-tune a suitable student LM with a smaller model size

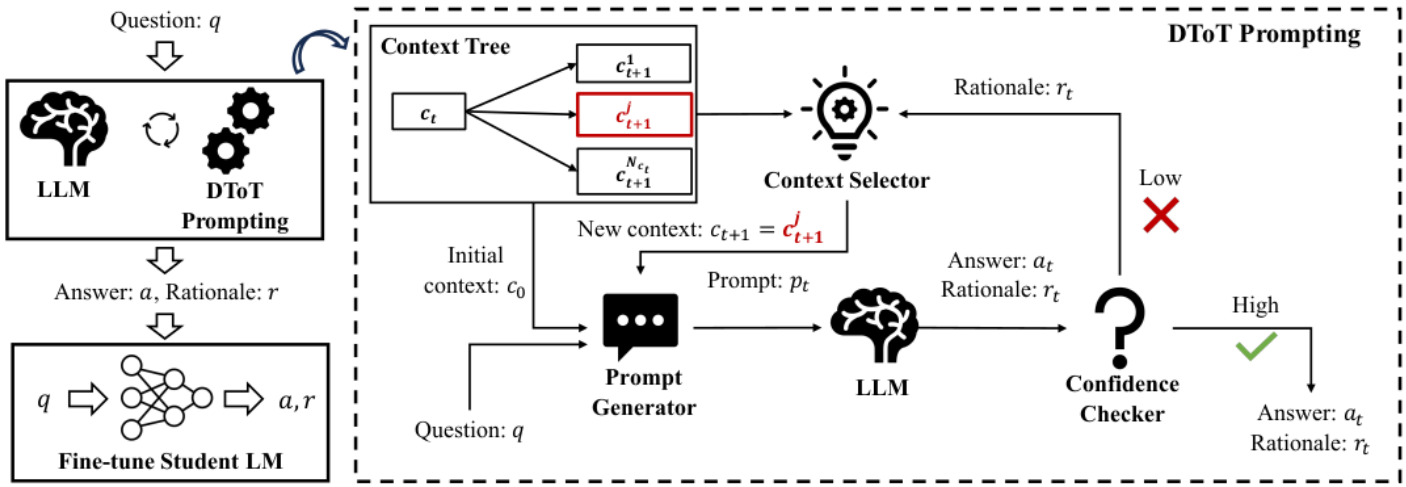


Figure 1: Illustration of BD-LLM

# Toxicity Detection with Generative Prompt-based Inference

- Yau et al. use a generation-based approach to classify a text $x = (x_1, x_2, \ldots, x_T)$.
- They claim that certain prompts steer the model towards generating toxic content, which they use to do so.
- Given such positive and negative prompts $y^p$ and $y^n$, they calculate the likelihood that $x$ is generated given $y \in \{y^p, y^n\}$ as:

$$s(y) = \sum_{t=1}^{T} log\ P_M(x_t|y, x_{<t})$$

- If $s(y^p) > s(y^n)$, then $x$ is classified as toxic.

# Good-old BERT

- BERT has long been used to classify texts.
- Even with the challenges posed by toxic content, we wish to experiment with the classic method of using BERT with a classifier head.

# Classification with Clustering

- This approach maintains a vector database of toxic examples.
- When we get a new input, we use the vector database to extract k-nearest-neighbours toxic examples.
- We then use these examples for k-shot prompting.
- For classification, we can either use a transformer with a classifier head or an API call to an LLM (for example, GPT) to get a "yes" or "no" output.

# Classification with Clustering

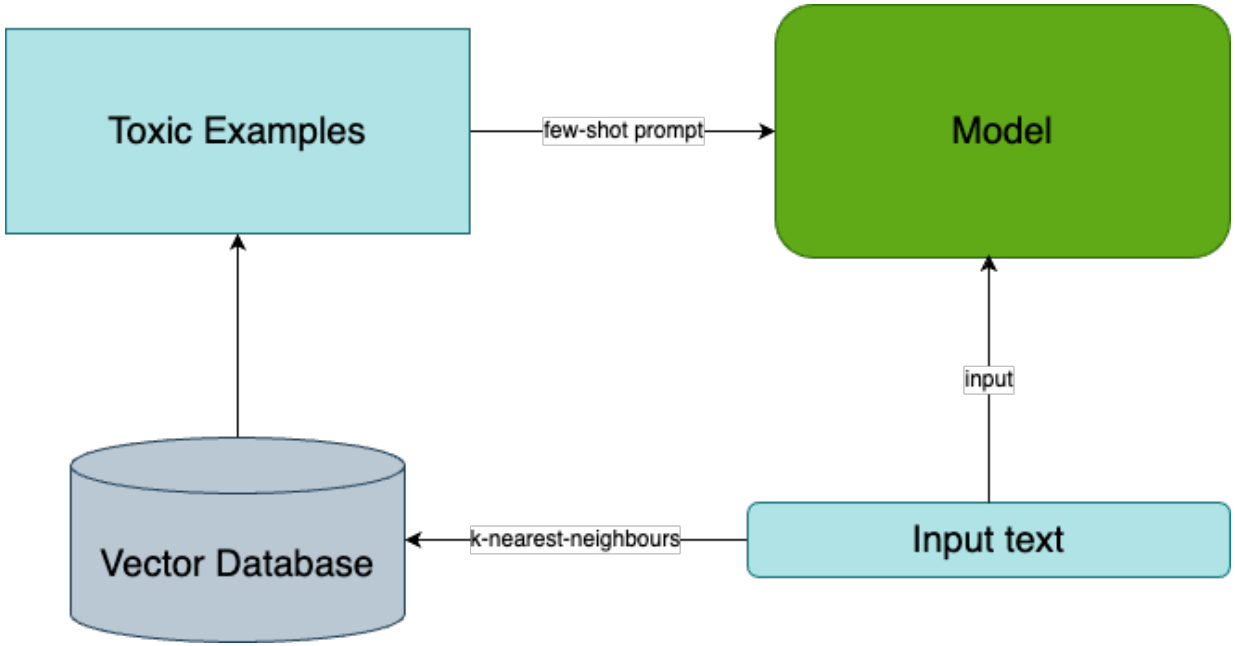


Figure 2: Architecture

# Thank you for listening!

Feel free to ask questions