
Toxic Comment Detection: Rule-Based Heuristics vs. Transformer Pipeline 😊

Alvin Kang (2025849087)

1 Introduction

This project provides a gentle introduction to designing simple **AI pipelines**. Rather than training large models or reading extensive research literature, you will:

- Choose a small, concrete problem solvable with an AI pipeline (e.g., text classification, retrieval, simple QA, image classification).
- Choose or collect a small dataset (e.g., from **datasets**).
- Implement a simple *naïve baseline* (e.g., rule-based or heuristic).
- Build an improved pipeline using existing pre-trained models.
- Evaluate both approaches using appropriate metrics.
- Reflect on what worked, what failed, and what you learned.

The emphasis is on the *process* of AI work: problem definition, pipeline design, evaluation, iteration, and writing. Your problem should be small enough to run comfortably on a single GPU (e.g., RTX 3090) or CPU.

Your tasks. Please replace all the sections to complete your report, starting from adding your project title, your name, and student ID above! Feel free to reorganize the sections. Then, submit a **public** github repository link that includes your code (including Jupyter notebook with results) and report **in PDF**, via LearnUs.

You can use tables and figures, and cite references using `\citet` ([1]) or `\citet` (Wei et al. [1]).

2 Task Definition

- **Task description:** The objective is to classify online comments as toxic or non-toxic.
- **Motivation:** Toxic Comment Detection is crucial for moderating online platforms and improving the quality of discussion boards in order to promote healthier user engagement.
- **Input / Output:** Input: A single text string or comment; Output: Binary label, labeling Toxic as 1 and Non-toxic as 0.
- **Success criteria:** I will know if the system is successful if the system is detecting a higher F1 score for the toxic comments, meaning that it can detect those comments.

3 Methods

This section includes both the naïve baseline and the improved AI pipeline.

3.1 Naïve Baseline

Implement a simple method that does not rely on heavy models. Examples include:

- Keyword-based text classification,
- Simple color/shape heuristics for image tasks,
- String-overlap-based retrieval.

In your report, explain:

- How the baseline works,
- Why it is considered naïve,
- Expected failure cases.

Your Baseline(TODO)

- **Method description:** If a comment contains a keyword from a predefined list of common profane words or insults it classifies as toxic, if it does not include the keyword it will be classified as non-toxic.
- **Why naïve:** It is naïve because the method ignores context, intent, and negation, only relying on keyword matching.
- **Likely failure modes:** It is likely to fail when the toxic language is indirect like a false negative comment, sarcastic, or uses words that are not part of the predefined list.

3.2 AI Pipeline

Design a small pipeline using one or more pre-trained models. Examples include:

- **Text:** embedding encoder + classifier, or a small transformer model,
- **Retrieval:** embedding model + nearest-neighbor search,
- **Vision:** pre-trained classifier (e.g., ViT-tiny).

A typical pipeline contains:

1. Preprocessing,
2. Embedding or representation,
3. Decision/ranking component,
4. Optional post-processing.

Fine-tuning large models is not required; inference-only usage is sufficient.

Your Pipeline (TODO)

- **Models used:** DistilBERT is a pre-trained transformer model for toxic comment classification.
- **Pipeline stages:** 1. Preprocessing: Lowercasing and tokenization of input comments, with truncation to 128 tokens. 2. Embedding/Representation: Encoding text using the DistilBERT transformer to obtain contextual representations. 3. Decision: Selecting the class with the highest predicted probability (toxic or non-toxic). 4. Post-processing: Applying a probability threshold of 0.5 to produce a binary output.

- **Design choices and justification:** DistilBERT was chosen for its strong performance and computational efficiency. Using a pre-trained model allows the pipeline to capture contextual meaning without requiring additional training, making it suitable for a lightweight and reproducible AI pipeline.

4 Experiments

4.1 Datasets

You may use a small public dataset (e.g., from `datasets`) or construct your own. In this section, describe:

- **Dataset source:** where it comes from.
- **Size:** number of examples used.
- **Splits:** how you divided train/validation/test.
- **Preprocessing:** e.g., tokenization, resizing, truncation, normalization.

Your Dataset Description(TODO)

- **Source:** Jigsaw Toxic Comment Classification data set, on Kaggle.
- **Total examples:** A subset of 1000 comments was used for this project.
- **Train/Test split:** The data set was randomly divided into 80 training data and 10 testing data.
- **Preprocessing steps:** All text was lower-cased prior to processing. Comments were tokenized using the tokenizer associated with the DistilBERT model and truncated to 128 tokens maximum. No additional filtering or cleaning was applied.

4.2 Metrics

Use at least one quantitative metric appropriate for your task:

- **Classification:** accuracy, precision, recall, F1,
- **Retrieval:** precision@k, recall@k,
- **Simple generation:** exact match, ROUGE-1.

It's worth considering how the metrics you select align with your tasks.

4.3 Results

Report:

- Metric values for baseline vs. pipeline,
- A results table,
- At least three qualitative examples.

Method	Metric 1	Metric 2 (optional)
Baseline	0.65	0.58
AI Pipeline	0.92	0.91

If you want to visualize results with any other than tables, refer to below links

- [Matplotlib official tutorial: Introduction to pyplot](#)

- Matplotlib example gallery (many bar/line/scatter plots with source code)
- Kaggle notebook: Matplotlib tutorial for beginners (interactive code)

5 Reflection and Limitations

Write approximately 6–10 sentences reflecting on:

- What worked better than expected,
- What failed or was difficult,
- How well your metric captured “quality”,
- What you would try next with more time or compute.

Your Reflection(TODO)

This project helped me realize how large the gap is between simple rule-based systems and modern language models. The AI pipeline performed better than I expected, particularly on comments where toxicity was implied rather than directly stated. Designing the naïve baseline was more difficult than anticipated, as it forced me to think carefully about how humans recognize toxicity versus how machines do. The keyword-based approach often felt insufficient when I tested it on real examples, especially those involving negation or subtle insults. Accuracy and F1-score were helpful in comparing overall performance, but they did not fully reflect the nuanced ways in which toxicity can appear in language. I also noticed that the pre-trained model sometimes struggled with sarcasm or ambiguous intent, reminding me that these models are not perfect. This limitation became more apparent as I examined individual prediction errors. With more time or computational resources, I would try fine-tuning the model and exploring richer evaluation metrics to better capture real-world content moderation challenges.

References

- [1] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations (ICLR)*, 2022. URL <https://openreview.net/forum?id=gEZrGCozdqR>.