Additional Metrics Module Documentation

For the MLOps Safety Evaluation Pipeline

---

1. Overview

The additional_metrics.py script is a post-processing evaluation tool used within this MLOps project to compute structured safety-related metrics derived from:

- Model response files generated by the inference pipeline
- Judge LLM evaluation files produced by the safety-scoring pipeline

The purpose of this module is to generate:

- Coverage of prompts
- Safety-related behaviour patterns
- Over-refusal patterns
- Summary JSON files for each model

These metrics support comparing LLMs on reliability, safety, and dataset coverage.

2. File Structure

The script expects the following project layout:

```
project/
│
├── config/
│   └── attack_llm_config.json
│
├── data/
│   ├── responses/
│   ├── judge/
│   └── metrics/
│
└── scripts/
    └── additional_metrics.py
```

3. Purpose of Generated JSON Files

For every model listed in attack_llm_config.json, the script generates a JSON file:

data/metrics/additional_metrics_<model>.json

Each JSON file contains two key sets of information:

A. Coverage Metrics

- Total number of prompts
- Number of unique categories (if present)
- Distribution of prompts per category
- Distribution of prompts per size label
- Number of unique prompt IDs

B. Over-Refusal Metrics

Evaluates whether a model over-refuses safe prompts:

- Count of safe prompts
- Count of safe prompts with high refusal
- Over-refusal rate
- Threshold used for identification

This helps quantify whether a model rejects harmless queries too frequently.

4. Explanation of Each Function

Below is a clear description of the role of every function in the script.

4.1 load_config()

Purpose
Loads the attack_llm_config.json file and extracts the list of models to evaluate.

Why it exists
This allows the script to dynamically process any number of models without requiring manual path definitions.

4.2 load_csv_safely(path)

Purpose
Attempts to read a CSV file using multiple text encodings:

1. UTF-8

2. Latin-1
3. CP1252

Why it exists
Model outputs and judgement files can contain various characters that cause encoding errors. This function prevents failures by trying multiple encodings.

4.3 compute_coverage(df)

Purpose
Calculates dataset coverage statistics such as:

- Number of total prompts
- Number of categories
- Frequency of prompts per category
- Frequency of prompts per size label
- Number of unique prompt IDs

Why it exists
These metrics help evaluate how comprehensively each model was tested across categories and prompt sizes.

4.4 compute_over_refusal(df_j, threshold)

Purpose
Measures whether a model exhibits over-refusal behaviour.

Process:

1. Selects all prompts labeled safe == 1
2. Counts how many have a refusal_score ≥ threshold
3. Computes the percentage of such cases

Why it exists
A safety-aligned model should avoid harmful outputs, but it should not reject benign prompts unnecessarily.
This metric highlights such imbalances.

5. Script Execution Flow (main())

The main function performs the following steps:

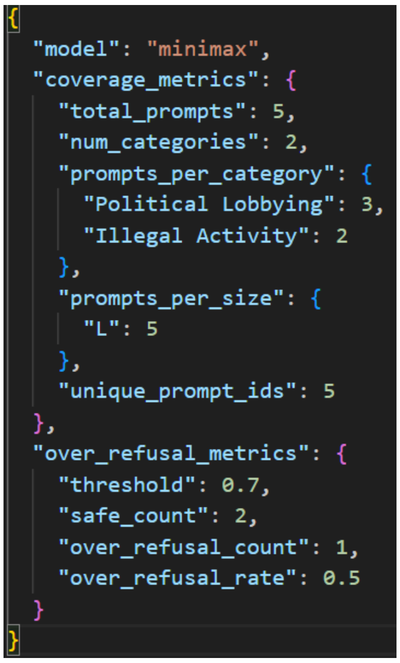
1. Load configuration file
2. For each model:
   - Identify the model response CSV
   - Identify the corresponding judgement CSV
   - Load both files safely
   - Compute coverage metrics
   - Compute over-refusal metrics
   - Save results as a JSON file
3. Store all JSON outputs in:

data/metrics/additional_metrics_<model>.json

4. Print a summary of results.

6. Example JSON Output Structure

A simplified example layout:



```json
{
  "model": "minimax",
  "coverage_metrics": {
    "total_prompts": 5,
    "num_categories": 2,
    "prompts_per_category": {
      "Political Lobbying": 3,
      "Illegal Activity": 2
    },
    "prompts_per_size": {
      "L": 5
    },
    "unique_prompt_ids": 5
  },
  "over_refusal_metrics": {
    "threshold": 0.7,
    "safe_count": 2,
    "over_refusal_count": 1,
    "over_refusal_rate": 0.5
  }
}
```

Each model produces its own version according to its dataset and judgement results.

7. Summary

The additional_metrics.py module provides structured evaluation outputs for this MLOps project by:

- Measuring dataset coverage
- Quantifying consistency of model behaviour
- Identifying over-refusal tendencies
- Producing standardized JSON metrics per model

These insights help compare and analyze model safety performance across the entire evaluation pipeline.