Final Project Proposal

YuChen (Jean) Lin `lin.4842@osu.edu`
Pakhi Chatterjee `chatterjee.197@osu.edu`
Aditya Pandey `pandey.172@osu.edu`

1 Problem Statement

Toxic comment detection models for Japanese are typically trained and evaluated on native-script text (kanji/kana). In practice, social text often appears in romaji (romanized Japanese) for ease of typing, stylistic effect, or to evade filters yet most open models are not evaluated for script invariance. If a model flags 最低だ as toxic but misses the semantically identical ”saiteida,” moderation quality and fairness suffer. Recent Japanese toxicity resources exist, but they focus on native script; a paired, cross-script robustness evaluation is missing. We target that gap by building a lightweight script-invariant toxicity pipeline and a paired testbed that measures label stability when only the script changes.

2 Motivation

Japan has tightened penalties for online insults in 2022 (can lead to up to one year imprisonment), so reliable moderation in Japanese is not just academic policy talk. Yet moderation quality is very uneven outside of English, with evidence of under-investment and missing language expertise across platforms, implying that harmful content can slip through when it is not written in the native language. In practice, rōmaji appears constantly because the standard Japanese IME accepts rōmaji keystrokes and converts them to kana/kanji, so Romanized surface forms are routine and should be treated as first-class inputs.

3 Research Gap

Japanese toxicity datasets such as LLM-jp's Japanese Toxicity Dataset (v2) support safer LLMs by providing strong native-script supervision but do not benchmark rōmaji robustness, and although we are considering additional datasets for more information, none has been finalized. Industry work on Japanese toxicity modeling explores efficient toxic language detection for real-world settings and underscores deployment needs but similarly overlooks script invariance. Dedicated studies on Roman Urdu hate speech demonstrate that romanization is treated as a real failure mode and provide a useful precedent [1], and a recent Nature report on multilingual toxicity detection highlights the necessity of script-agnostic robustness [2]. Yet no equivalent work focuses on Japanese rōmaji. This project fills that gap by creating a Japanese rōmaji-robust toxicity benchmark. Consequently, there is a critical need for establishing a standardized benchmark to measure how well toxicity classifiers handle Japanese text when presented in romanized form.

4 Technical Resources and Datasets

4.1 Models

- BERT-base-multilingual-cased (Hugging Face)
- XLM-RoBERTa-base (Hugging Face)

4.2 Training Datasets

- inspection-ai/japanese-toxic-dataset [4]: Contains native Japanese text with toxicity level annotations

- llm-jp-toxicity-dataset [3]: Japanese toxicity dataset with LLM-provided quality assessments and statistics
- Japanese LLM Safety dataset [5]: Comprehensive safety evaluation dataset for Japanese language models

4.3 Romanization Pipeline

pykakasi (v2.2+) [6] for reliable Japanese-to-romaji conversion using Hepburn romanization standard

4.4 Compute Resources

- Single GPU instance (e.g., NVIDIA V100, 16 GB)
- Python 3.9, PyTorch, Hugging Face Transformers

5 Proposed Approach

5.1 Data Preprocessing and Validation

- Convert Japanese datasets to romaji using pykakasi with Hepburn romanization
- Manually inspect 200 random samples; record and fix conversion errors (e.g., loanwords, long vowels)
- Create train/validation/test splits (70/15/15) maintaining toxicity level distribution
- Produce summary statistics: token counts, class distributions

5.2 Model Fine-tuning

- Fine-tune BERT-base-multilingual and XLM-RoBERTa-base on romanized Japanese toxicity data
- Implement stratified sampling on Romanized Japanese to address potential class imbalance
- Use standard hyperparameters: learning rate (e.g. 2e-5), batch size (e.g. 16), (e.g. 3-5) epochs with early stopping
- Apply cross-validation to ensure robust performance estimates

6 Evaluation

6.1 Classification Performance

- F1_native: Precision, Recall, and F1-score on the held-out native-script test set (per level and macro-averaged)
- F1_romaji: Precision, Recall, and F1-score on the held-out romaji test set (per level and macro-averaged)

6.2 Script-Invariance Gap

- For each toxicity level:

  $$\Delta F_1 = F_{1,\text{native}} - F_{1,\text{rōmaji}}$$

  report mean ΔF1 and 95% confidence interval
- Label Consistency Rate: % of test instances whose predicted label is identical in native and romaji forms (Flip rate = 1 - Consistency).

6.3 Statistical Significance

- McNemar's Test: Conduct McNemar's test on paired native versus romaji predictions to determine if the difference in error rates is statistically significant

6.4 Robustness to Romanization Variants

- For each common perturbation (long-vowel representation, gemination, numeric leet), compute

$$\Delta F_{1,\text{variant}} = F_{1,\text{rōmaji}} - F_{1,\text{variant}}$$

  report mean and standard deviation

6.5 Error Analysis on Romanized Inputs

- High-ΔF1 Case Review: Manually examine the top 50 samples with the largest native–romaji F1 discrepancy to identify systematic romanization failure modes

6.6 Efficiency Metrics for Deployment (Optional)

- Inference latency (ms/sample) and throughput (samples/s) on GPU and CPU
- Peak GPU memory usage during batch inference

7 Expected Outcomes

- A paired, script-controlled benchmark for Japanese toxicity detection
- A drop-in training recipe (romanization + consistency loss) that reduces flip rate without sacrificing F1
- Reproducible code and dataset construction scripts for future Japanese moderation research (and portability to other languages with romanization)

8 Timeline

- **W1:** Set up data access + build/verify rōmaji conversion pipeline (paired prototype)
- **W2:** Train baselines (native, rōmaji, mixed) and wire up F1/AUROC/Flip Rate
- **W3:** Implement dual-view consistency loss and run first script-invariant model
- **W4:** Hyperparameter sweep (λ, LoRA rank) and apply temperature scaling (ECE/Brier)
- **W5:** Run ablations and slice/error analysis (emoji, elongations, mixed-script, length)
- **W6:** Finalize results, package code/data, and prepare write-up + slides

References

[1] Hammad Rizwan, Muhammad Haroon Shakeel, and Asim Karim. 2020. Hate-Speech and Offensive Language Detection in Roman Urdu. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2512–2522, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.197

[2] Ashiq, W., Kanwal, S., Rafique, A. et al. Roman urdu hate speech detection using hybrid machine learning models and hyperparameter optimization. Sci Rep 14, 28590 (2024). https://doi.org/10.1038/s41598-024-79106-7

[3] LLM-jp Consortium (2024). *Japanese Toxicity Dataset v2*. https://gitlab.llm-jp.nii.ac.jp/datasets/llm-jp-toxicity-dataset-v2

[4] Inspection AI (2024). *Japanese Toxic Dataset*. https://github.com/inspection-ai/japanese-toxic-dataset

[5] Japanese LLM Safety Research Group (2024). *Japanese LLM Safety Dataset*. arXiv preprint arXiv:2506.02372. https://doi.org/10.48550/arXiv.2506.02372

[6] pykakasi Development Team (2024). *pykakasi: Japanese text transliteration library*. https://pypi.org/project/pykakasi

[7] Oikawa, Yuto, Yuki Nakayama, and Koji Murakami. 2022. *A Stacking-based Efficient Method for Toxic Language Detection on Live Streaming Chat.* In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 571–578, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.emnlp-industry.58