Proposal: Fast and Resource-Efficient Safety Filters for LLM Outputs

Course: Generative AI
Semester: Spring 2026

Team Members:

- Arthur Babkin
- Alexander Malyy

1. Problem Statement & Motivation

Large language models are commonly deployed with **post-generation safety filters** that detect harmful or toxic text. In real-world systems, these filters are often applied at scale and under strict latency and resource constraints.

The core problem addressed in this project is the **trade-off between speed and resource usage versus detection quality**:

- Lightweight models are fast and cheap but may miss subtle harmful content.
- More expressive models achieve higher quality but increase latency, memory usage, and compute cost.

This project aims to systematically compare **classical and neural safety filters** under a unified evaluation protocol, focusing on how **model capacity, preprocessing, and decision rules affect inference speed, memory usage, and detection quality**.

The problem is important because:

- Safety filters are often on the critical path of LLM inference.
- Even small latency increases can be unacceptable in production systems.
- Resource-efficient safety mechanisms are essential for CPU-only or edge deployments.
- Russian-language content adds complexity due to morphology, slang, and limited coverage in existing models.

2. Data

Data sources

- Publicly available datasets for toxic or harmful text classification with **binary labels** (`safe` / `unsafe`).
- Multilingual data will be used, with a specific evaluation focus on **Russian-language samples**.
- Since there are a lot of datasets available, we will concatenate several ones and transform multi-label datasets into binary ones.
- Some examples: multilingual + multilabel, russian + binary, russian + multilabel, english + binary.

License and usage

- Only datasets with clear research-friendly licenses will be used.
- Dataset licenses and usage constraints will be documented in the final report.

Preprocessing steps

- Basic text normalization (lowercasing, whitespace cleanup).
- Fixed train/validation/test splits with recorded random seeds.
- Two preprocessing configurations:
  - **No defense:** minimal normalization.
  - **Deobfuscation defense:** simple deterministic rules (e.g., collapsing spaced characters, normalizing repeated letters, basic leetspeak mapping).

A separate **obfuscation stress set** will be generated by applying deterministic transformations to the test split. This stress set will be used **only for evaluation**.

3. Baseline Plan

Initial implementation

The first system implemented will be a **fast, CPU-friendly baseline**:

- **Model:** TF-IDF features + Logistic Regression
- **Task:** Binary classification (`safe` vs `unsafe`)
- **Motivation:**
  - Very low inference latency.
  - Minimal memory footprint.
  - Strong classical baseline for text classification.

Capacity comparison

A **lightweight Transformer-based classifier** will then be implemented and evaluated under the same conditions.

The comparison will explicitly measure:

- Inference latency per sample
- Memory usage during inference
- Detection quality under clean and obfuscated inputs

4. Evaluation Plan

Quality metrics

- Precision, Recall, F1-score
- Precision–Recall AUC (PR-AUC)
- Confusion matrix at a fixed operating point

Speed and resource metrics

- Average inference latency per sample (CPU and, if available, GPU)
- Throughput (samples per second)

- Peak memory usage during inference
- Model size on disk

Evaluation protocol

- All models trained on identical training splits.
- Thresholds selected on the validation set using a fixed rule (e.g., max F1).
- Evaluation performed on:
    - Clean test data
    - Obfuscated test data

Comparative analysis

Results will be reported as:

- Quality vs latency trade-off curves
- Quality vs memory usage comparisons
- Relative slowdowns and quality gains compared to the TF-IDF baseline
- Where applicable, comparison of our metrics to published or leaderboard results from Kaggle competitions that use the same datasets as in the **Data sources** section.

5. Risks

Compute limitations

- Transformer models may require limited GPU access.
- Mitigation: restrict model size and always include a CPU-only baseline.

Data quality concerns

- Labels may be noisy or ambiguous.
- Mitigation: focus on comparative trends rather than absolute scores.

Potential failure modes

- TF-IDF baseline missing semantic or implicit harmful content.
- Transformer model being too slow for practical deployment.
- Deobfuscation rules increasing false positives.

6. Ethics & Safety Considerations

Misuse potential

- Safety filters can be misused for excessive censorship.
- This project focuses on performance characteristics, not policy decisions.

Privacy and data sensitivity

- Datasets may contain offensive or harmful language.
- No private or personally identifiable information will be collected or generated.

7. Timeline

- **Week 2–3:** Dataset selection, preprocessing, proposal submission
- **Week 4–5:** Implement TF-IDF + Logistic Regression baseline and speed evaluation
- **Week 6:** Midterm checkpoint (baseline results and initial analysis)
- **Week 7–8:** Implement Transformer classifier and capacity comparison
- **Week 9:** Robustness and deobfuscation experiments
- **Week 10–12:** Final analysis, report writing, reproducibility checks, demo preparation

8. References

1. Jigsaw Toxic Comment Classification Challenge (Kaggle). *Toxic comment classification dataset (multilingual, multilabel)*. https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge
2. Blackmoon. *Russian Language Toxic Comments Dataset (binary labels)*. https://www.kaggle.com/datasets/blackmoon/russian-language-toxic-comments
3. Semiletov, A. *Toxic Russian Comments Dataset (multilabel)*. https://www.kaggle.com/datasets/alexandersemiletov/toxic-russian-comments
4. Abusaqer, M. *Combined Hate Speech Dataset (English, binary)*. https://www.kaggle.com/datasets/mahmoudabusaqer/combined-hate-speech-dataset
5. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. NAACL-HLT, 2019.
6. Liu, Y. et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv:1907.11692, 2019.
7. Schmidt, A., & Wiegand, M. *A Survey on Hate Speech Detection using Natural Language Processing*. Proceedings of the 5th International Workshop on Natural Language Processing for Social Media, 2017.
8. Fortuna, P., & Nunes, S. *A Survey on Automatic Detection of Hate Speech in Text*. ACM Computing Surveys, 2018.