USENIX Security '25 Artifact Appendix: HATEBENCH: Benchmarking Hate Speech Detectors on LLM-Generated Content and Hate Campaigns

Xinyue Shen[1] Yixin Wu[1] Yiting Qu[1] Michael Backes[1] Savvas Zannettou[2] Yang Zhang[1]
[1]*CISPA Helmholtz Center for Information Security* [2]*Delft University of Technology*

A Artifact Appendix

A.1 Abstract

In this paper, we propose HateBench, a framework for benchmarking hate speech detectors on LLM-generated hate speech. We first construct a hate speech dataset of 7,838 samples generated by six widely-used LLMs covering 34 identity groups, with meticulous annotations by human labelers. We then assess the effectiveness of eight representative hate speech detectors on the LLM-generated dataset. Our results show that while detectors are generally effective in identifying LLM-generated hate speech, their performance degrades with newer versions of LLMs.

We also reveal the potential of LLM-driven hate campaigns, a new threat that LLMs bring to the field of hate speech detection. By leveraging advanced techniques like adversarial attacks and model stealing attacks, the adversary can intentionally evade the detector and automate hate campaigns online. The most potent adversarial attack achieves an attack success rate of 0.966, and its attack efficiency can be further improved by 13-21x through model stealing attacks with acceptable attack performance.

A.2 Description & Requirements

This artifact includes code to generate the main result tables in the paper, including:

- Table 3: Performance on LLM-generated samples.
- Table 4: F1-score on LLM-generated and human-written samples.
- Table 6: Performance of adversarial hate campaign
- Table 8: Performance of model stealing attacks.
- Table 9: Performance of stealthy hate campaign with black-box attacks.
- Table 10: Performance of stealthy hate campaign with white-box gradient optimization.

Tables 3 and 4 can be executed directly on a local PC without requiring a GPU environment. For other experiments, we recommend using an environment with NVIDIA GeForce RTX 3090 or more powerful GPUs, such as the RTX 4090 or A100. The results presented in this paper were obtained using an NVIDIA GeForce RTX 3090.

A.2.1 Security, privacy, and ethical concerns

Reproducing Table 3 and Table 4 does not pose any security risks. However, to reproduce the results of the hate speech campaign (Table 6, 8, 9, 10), it is likely to introduce security and ethical concerns, as it involves attacking commercial models such as the Perspective API and OpenAI Moderation API. To mitigate these risks, we recommend that evaluators reproduce the results on the open-source model TweetHate, which is also the default target model provided in the following steps. This ensures that evaluator results do not affect the functionality of commercial models.

A.2.2 How to access

The artifact is available on Zenodo: https://zenodo.org/records/14840447. Given the ethical concerns surrounding our code, which includes attacks against real-world systems, we host these artifacts on Zenodo with the request-access feature enabled.

A.2.3 Hardware dependencies

The artifact requires environments with GPUs. We recommend using an environment with NVIDIA GeForce RTX 3090 or more powerful GPUs, such as the RTX 4090 or A100. The results presented in this paper were obtained using an NVIDIA GeForce RTX 3090.

A.2.4 Software dependencies

All our experiments are tested in a conda environment on Ubuntu 20.04.6 LTS with Python 3.9.0. We provide detailed instructions for evaluators to set up this conda environment (see subsection A.3).

A.2.5 Benchmarks

Two datasets are used in the main experiments: 1) HateBench-Set, the manually-annotated dataset generated by LLMs. 2) MHS dataset[1] as human-written samples.

A.3 Set-up

A.3.1 Installation

Download the artifact from Zenodo, decompress it, and enter the main project directory. Then follow the below commands to build the environment.

```
conda create -n hatebench python=3.9.0
conda activate hatebench
pip install -r requirements.txt
```

Enter the PYTHON environment to download the necessary package.

```
import nltk
nltk.download('averaged_perceptron_tagger_eng')
exit()
```

A.3.2 Basic Test

```
python basic_test.py
```

Output should be

```
Transformers loaded successfully!
Detoxify works! output: {...}
TextAttack module initialized successfully!
OpenAttack loaded successfully!
Pandas & NumPy work!
Basic test completed!
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We show that hate speech detectors perform inconsistently on LLM-generated and human-written samples. This is proven by the experiment (E1, E2) described in Section 4, with the results shown in Tables 3 and 4.

(C2): Adversarial hate campaigns can bypass detectors. This is proven by the experiment (E3) described in Section 5.4, whose results are reported in Table 6.

(C3): Model stealing can be used to enhance stealthy hate campaigns. This is proven by the experiment (E4, E5) described in Section 5.5, whose results are illustrated in Tables 8, 9, and 10.

[1] https://huggingface.co/datasets/ucberkeley-dlab/measuring-hate-speech.

A.4.2 Experiments

(E1): Performance on LLM-generated samples *[1 compute-minute]:*
How to: Run the script to get the results.
Preparation: None
Execution: Run the following command in the terminal from the project's root directory.

```
python measurement/
    calculate_detector_performance.py
```

Results: The results will be printed directly on the terminal. It should be similar to Table 3.

(E2): F1-score on LLM-generated and human-written text. *[2 compute-minutes]:*
How to: Run the script to calculate the performance of hate speech detectors.
Preparation: None
Execution: Run the following command in the terminal from the project's root directory.

```
python measurement/
    calculate_detector_LLM_performance.py
```

Results: The results will be printed directly on the terminal. It should be similar to Table 4.

(E3): Adversarial hate campaign *[30 compute-minutes]:*
How to: Execute the attack scripts to reproduce the results of TweetHate presented in Table 6. Results will be automatically stored in the *./logs/* directory. The naming convention for the log files is *adv_hate_campaign_{target_model}_{attack}.log*. For example, to check the results of *TextFooler* attack on *TweetHate* model, refer to the log file named *adv_hate_campaign_TweetHate_TextFooler.log* and the results are printed at the end of the log.
Preparation: None
Execution: Run the script.

```
cd hate_campaign
bash scripts/run_adversarial_hate_campaign.sh
```

Results: The results are expected to be comparable to those for TweetHate in Table 6. Note that minor deviations may arise due to the inherent stochasticity of machine learning algorithms and differences in GPU performance.

(E4): Steal the target model *[2 compute-hours]:*
How to: Run the script and check the results.
Preparation: None
Execution: Run the script.

```
nohup python model_stealing.py --target_model
    TweetHate --surrogate_model roberta  > ./
    logs/TweetHate_roberta.log &
nohup python model_stealing.py --target_model
    TweetHate --surrogate_model bert  > ./
    logs/TweetHate_bert.log &
```

Results: The results are expected to be comparable to those for TweetHate in Table 8.

(E5): Stealthy hate campaign (black-box + white-box) *[60 compute-minutes]:*

How to: Execute the attack scripts to reproduce the results of TweetHate presented in Tables 9 and 10. Results will be automatically stored in the *./logs/* directory.

Preparation: None

Execution: Run the script.

```
bash scripts/run_stealthy_hate_campaign.sh
```

Results: The results are expected to be comparable to those for TweetHate in Tables 9 and 10. Attack *textfooler* refers to the black-box attack (Table 9). Attack *textfooler_gradient* is the white-box attack (Table 10). The results are printed at the end of the log.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.