Tailoring In-Context Learning Techniques for Definition-Based Hate Speech Detection in Large Language Models

Parham Bateni[1]

Supervisor(s): Prof. Pradeep Murukannaiah[1]**, Urja Khurana**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2026

Name of the student: Parham Bateni
Final project course: CSE3000 Research Project
Thesis committee:
Responsible Professor: Prof. Pradeep Murukannaiah,
Supervisor: Urja Khurana,
Examiner: Prof. Cynthia Liem

Abstract

Hate speech lacks a single agreed definition across legal, social, and benchmark contexts, yet instruction-tuned large language models (LLMs) are increasingly used for hate speech detection. While recent work has explored definition-aware prompting, it remains unclear how different definitions interact with few-shot prompting strategies and model capacity. We investigate whether zero-shot and few-shot in-context learning can align LLMs with dataset-specific hate speech definitions without fine-tuning. Using the HateCheck benchmark, we evaluate three models (Gemma-2-2B, Llama-3.2-3B, and Qwen2.5-3B) under three definition settings (no definition, author-provided text, and structured criteria-based definition) and four prompting strategies (zero-shot and three few-shot variants). Results show that explicit definitions do not reliably improve performance and can sometimes reduce it. Furthermore, few-shot prompting is generally more effective, with the strongest performance often achieved by retrieving semantically similar examples for each query and including them in the prompt. In addition, higher-capacity models benefit more from richer prompts, whereas the smallest model frequently degrades as prompt complexity increases. Overall, definition wording, exemplar selection, and model capacity interact strongly and should be tuned jointly rather than considered in isolation.[1]

1 Introduction

Large language models (LLMs), particularly instruction-tuned variants that follow natural language task specifications, are increasingly deployed in everyday applications [1, 2, 3]. Their strong language understanding and ability to adapt to tasks through prompting make them an attractive approach for content moderation and hate speech detection. However, applying them to this task remains challenging because hate speech is not universally defined and varies across legal, social, and dataset-specific contexts [4].

This lack of a consistent definition creates a key research gap. In particular, it remains unclear how explicitly encoding dataset-specific hate speech definitions in prompts, together with in-context examples, affects classification performance. Moreover, it is not well understood whether prompting-based methods can serve as an effective alternative to costly fine-tuning.

To evaluate this alternative, it is useful to distinguish prompting-based techniques from fine-tuning. In-context learning allows a model to infer a task from natural language instructions and examples provided in its input context [5], while prompt engineering refers to the design of such instructions and exemplars without updating model parameters [6]. In contrast, fine-tuning adapts a model to a task by training on labeled data and updating its parameters, which is typically more computationally expensive and time-consuming. Whether prompting-based approaches can achieve comparable alignment when dataset-specific definitions are included in the prompt remains an open question.

Recent work has begun to explore this question in zero-shot settings [7, 8], where models receive task instructions and definitions but no labeled exemplars. However, the few-shot setting, in which labeled examples are additionally included in the prompt, and its interaction with model capacity, remain largely underexplored. We investigate the following research question:

How do zero-shot and few-shot prompting affect LLM performance in hate speech detection when prompts are enriched with dataset-specific definitions?

To answer this question, we break it down into three sub-questions:

- **RQ1:** Does injecting explicit hate speech definitions improve performance compared to generic prompting?
- **RQ2:** Which prompting strategy (zero-shot or few-shot) is more effective under definition-aware settings?
- **RQ3:** How does performance vary across instruction-tuned LLMs, and how does model choice interact with definition and prompting settings?

To address these questions, we conduct a factorial study on HateCheck [9], a functional benchmark for hate speech detection containing 3,728 annotated instances. The dataset is particularly suitable for our analysis because it comprises 29 controlled functionalities designed to test specific linguistic and pragmatic phenomena, including negation and contrastive non-hateful constructions, allowing for systematic evaluation beyond surface-level cues.

We compare three LLMs (Gemma-2, Llama-3.2, and Qwen2.5) under three definition settings (no definition, the original dataset definition, and a structured formulation based on the Hate Speech Criteria framework [10]) and four prompting strategies (zero-shot and three few-shot exemplar selection methods). Model performance is evaluated using macro F1 on a binary classification task (hateful versus non-hateful).

Our main findings are threefold. First, explicit hate speech definitions often degrade performance instead of improving it. Second, few-shot prompting generally outperforms zero-shot prompting. Third, performance varies substantially across LLMs, likely driven by differences in model capacity and architecture.

This paper makes the following contributions:

- We study how dataset-specific definitions influence prompt-based hate speech detection.
- We systematically compare zero-shot and few-shot prompting under definition-aware settings.
- We analyze the interaction between definition formulation, exemplar selection, and model performance.

[1] *Disclaimer:* This paper contains examples of hateful and abusive language. Reader discretion is advised.

The remainder of this paper is organized as follows. Section 2 reviews related work. Sections 3 and 4 present the methodology and few-shot prompting strategies, respectively. Section 5 reports and discusses the results. Finally, we discuss limitations and future work, followed by the conclusion.

2 Related Work

Definition of Hate Speech Hate speech has been defined in multiple ways across legal and academic contexts and remains difficult to standardize [4]. Matsuda et al. [11] describe it as abusive or assaultive speech targeting historically oppressed groups, while the Oxford English Dictionary defines it as abusive or threatening language expressing prejudice against a group [12]. In contrast, EU legislation focuses on public incitement to violence or hatred against groups defined by protected characteristics [13]. Although these definitions overlap in targeting groups, they differ in scope and interpretation across jurisdictions, which leads to inconsistencies in annotation and dataset design. For instance, the denial of genocides or mass atrocities may be classified as hate speech in several European jurisdictions, whereas the same expression is generally protected under the First Amendment in the United States. Hate speech is therefore better understood as a context-dependent concept rather than a fixed linguistic category.

Structured Representations of Definitions To handle the contextual variability of hate speech, several structured representations have been proposed. The Hate Speech Criteria (HSC) framework [10] breaks hate speech down into fine-grained dimensions such as target group, dominance, perpetrator role, type of negative reference, and possible consequences. Similarly, Melis et al. [8] introduce a modular taxonomy of concepts such as communication form, targets of hostility, protected attributes, and consequences that can be combined to represent different definitions from the literature.

Hate Speech Detection Hate speech detection has been widely studied in Natural Language Processing (NLP), evolving from rule-based systems and classical machine learning methods [14, 15] to neural and Transformer-based architectures that capture richer contextual nuances [16, 17, 18]. More recently, instruction-tuned LLMs have demonstrated strong performance in zero-shot classification through prompt-based reasoning, substantially reducing the need for task-specific fine-tuning [7]. However, model performance remains sensitive to dataset-specific definitions of hate speech [10, 14].

Evaluation frameworks have attempted to address this definitional sensitivity in different ways. DefVerify [19] evaluates whether hate speech models reflect the definitions underlying their training datasets. The framework decomposes dataset definitions into interpretable Hate Speech Criteria (HSC) aspects and measures model performance on diagnostic examples corresponding to those aspects, identifying mismatches between intended definitions and model behavior. However, it primarily evaluates fine-tuned models rather than prompting-based approaches. In contrast, Melis et al. [8] examine prompt-based approaches by exploring whether incorporating modular definitions improves performance in a zero-shot setting on the HateCheck dataset, but do not consider few-shot scenarios.

Key Research Gap Despite these developments, it remains unclear whether precise definition alignment can be achieved through prompting alone, without relying on computationally expensive fine-tuning. This question is especially open when definition format, exemplar selection, and model capacity are varied jointly. We address this gap by systematically integrating dataset-specific definitions into prompts and comparing zero-shot and few-shot strategies across instruction-tuned LLMs on HateCheck.

3 Methodology

Given a hate speech dataset, we extract its definition in different formats and inject it into prompts together with labeled examples (exemplars) using in-context learning. We then evaluate the resulting prompts across three models and compare predictions against dataset labels. Figure 1 illustrates this pipeline, and the following subsections describe each component in detail.

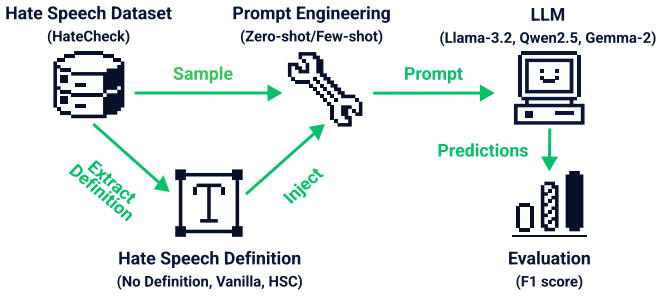


Figure 1: Overview of the pipeline for studying dataset-specific definitions in prompt-based hate speech detection. Definitions are extracted from the dataset and injected into prompts together with a query example. The resulting model predictions are compared against the ground-truth dataset labels for evaluation.

3.1 Hate Speech Dataset

We evaluate all experiments on HateCheck, a functional benchmark for hate speech detection. We select it because its contrastive non-hateful cases are deliberately constructed to resemble hateful ones. Correct classification therefore depends on the precise boundary of the hate speech definition, which is exactly the sensitivity our definition-aware study targets. The dataset contains 3,728 labeled English test cases, including 2,563 hateful and 1,165 non-hateful examples generated from manually designed templates.

HateCheck organizes its test cases into 29 functionalities, each designed to evaluate whether a model can correctly distinguish hateful from non-hateful text in a specific scenario. Most examples are generated from manually designed templates containing placeholders (e.g., `[IDENTITY_P]`) that are filled with different protected-group identifiers. Eighteen functionalities cover distinct hate forms such as slurs and threats; the remaining eleven are contrastive non-hate cases, including negations and counter-speech.

For example, functionality F1 evaluates hate expressed through strong negative emotions using templates such as *"I hate [IDENTITY_P]."*, which yields instances like *"I hate immigrants."* (hateful). In contrast, functionality F15 evaluates non-hateful negated statements using templates such as *"Let me be clear: I don't hate [IDENTITY_P]."*, yielding *"Let me be clear: I don't hate immigrants."* (non-hateful). Despite their similarity, the two examples belong to different classes.

3.2 Hate Speech Definitions

Prompts optionally include an explicit hate speech definition derived from the dataset. We compare three definition settings, chosen to span the range from no explicit guidance to a free-text definition and a fully structured one:

1. **No Definition:** no definition is included in the prompt.
2. **Vanilla:** the original textual definition provided by HateCheck authors.
3. **HSC:** a structured definition based on the Hate Speech Criteria (HSC) framework, consisting of the following dimensions:
   - **Target group:** which groups are considered potential targets of hate speech (e.g., gender, race, religion, sexual orientation).
   - **Dominance:** the social status of target groups, i.e., whether dominant groups may also be considered potential targets (e.g., white people, men).
   - **Perpetrator characteristics:** whether speaker identity or role affects the label (e.g., societal role, membership in the target group).
   - **Negative reference:** the type of explicit negative reference related to group membership (e.g., stereotype, group characteristic, slur).
   - **Potential consequences:** the potential effects an utterance may incite or imply (e.g., violence, hate, discrimination).

The complete extracted definition texts for each setting are provided in Appendix A.

3.3 Prompt Engineering

Given a hate speech definition and dataset examples, we construct the prompts using one of the following strategies:

1. **Zero-shot:** the model receives a task instruction and, when applicable, the dataset definition:

   > Detect hate speech based on the following definition:
   > {Dataset Hate Speech Definition}

2. **Few-shot:** the prompt additionally includes labeled in-context examples sampled from the dataset and balanced by class:

   > Detect hate speech based on the following definition and in-context examples:
   > {Dataset Hate Speech Definition}
   >
   > $\{\text{Example}_1\} \rightarrow$ non-hateful
   > $\{\text{Example}_2\} \rightarrow$ hateful
   > $\{\text{Example}_3\} \rightarrow$ non-hateful
   > ...

The few-shot examples are interleaved in the prompt by alternating between the two classes (non-hateful, hateful, non-hateful, ...), rather than listing all examples from one class before the other. This design choice is motivated by the known sensitivity of in-context learning to example order [20], where model predictions can be heavily influenced by contiguous blocks or recently seen labels. By alternating classes, both labels remain consistently present throughout the context, reducing the risk that the model latches onto spurious ordering patterns or local label bias. These specific few-shot exemplar selection strategies are elaborated in Section 4; the exact number of exemplars per class and the fixed exemplar sets are reported in Appendix E; and the full, unpopulated prompt templates are provided in Appendix B.

Finally, the models are instructed to output predictions in a rigid format:

> PREDICTION: {"hateful" or "non-hateful"}

Invalid responses trigger up to four additional generation attempts; samples that still fail to produce a valid prediction after all attempts are excluded from evaluation.

3.4 Large Language Models

We evaluate three instruction-tuned open-weight LLMs of comparable scale: `Llama-3.2-3B-Instruct` [21] (∼3.2 billion parameters), `Qwen2.5-3B-Instruct` [22] (∼3.1 billion parameters), and `Gemma-2-2B-Instruct` [23] (∼2.6 billion parameters). Instruction-tuned variants are selected because they are optimized to follow complex textual guidelines, which aligns with our prompt-based evaluation framework. We constrain our selection to models with a number of parameters in a tight 2 to 3 billion range, firstly because of computational constraints and secondly because downstream performance differences can be attributed to architecture and prompting rather than to disparities in scale. Further inference settings and model access details are described in Appendix D.

3.5 Evaluation

Predictions are compared against the gold labels using a binary confusion matrix over the `hateful` and `non-hateful` classes. Performance is measured using macro F1, computed as the unweighted mean of per-class F1 scores. We use macro F1 rather than accuracy because the dataset is class-imbalanced; accuracy is dominated by performance on the majority class, whereas macro F1 gives equal weight to hateful and non-hateful detection.

4 Few-Shot Prompting Strategies

As explained in Section 3.3, few-shot prompting lets LLMs learn from a small number of examples provided in the prompt context. However, the choice of examples is non-trivial and can affect model performance. Accordingly, we explore three exemplar-selection strategies for choosing examples from HateCheck, corresponding to the `Few-shot Random`, `Few-shot Diverse`, and `Few-shot Nearest` settings.

4.1 Few-shot Random

Random sampling is the simplest strategy for selecting examples to include in the prompt. We randomly sample $n/2$ examples from each class (hateful and non-hateful), where n is the total number of in-context examples included in the prompt. This approach is straightforward to implement but does not guarantee that the selected examples are diverse or representative of different aspects of the task.

4.2 Few-shot Diverse

Diverse sampling aims to select examples that are diverse from one another. To achieve this, we first represent each example using the `nomic-embed-text-v1.5` sentence embedding model [24], which maps sentences into dense vector representations of 768 dimensions. These embeddings are trained on large corpora and capture the semantic meaning of the input text.

Using these embeddings, we separately process each class of examples (hateful and non-hateful). For each class, we apply k-means clustering [25], which iteratively assigns each example to the nearest centroid and updates centroids until convergence, partitioning the examples into $n/2$ clusters. We compute the distance between two embeddings as Euclidean distance on L_2-normalized vectors. On unit-normalized vectors, this is equivalent to clustering by cosine similarity: minimizing squared Euclidean distance maximizes cosine similarity, as shown in Equation 1:

$$\begin{aligned}\arg\min_{\mathbf{e}_b} \|\hat{\mathbf{e}}_a - \hat{\mathbf{e}}_b\|_2^2 &= \arg\min_{\mathbf{e}_b} \left(\|\hat{\mathbf{e}}_a\|_2^2 + \|\hat{\mathbf{e}}_b\|_2^2 - 2 \cdot \hat{\mathbf{e}}_a \cdot \hat{\mathbf{e}}_b\right) \\ &= \arg\min_{\mathbf{e}_b} \left(2 - 2 \cdot \frac{\mathbf{e}_a \cdot \mathbf{e}_b}{\|\mathbf{e}_a\|_2 \|\mathbf{e}_b\|_2}\right) \\ &= \arg\max_{\mathbf{e}_b} \frac{\mathbf{e}_a \cdot \mathbf{e}_b}{\|\mathbf{e}_a\|_2 \|\mathbf{e}_b\|_2} \\ &= \arg\max_{\mathbf{e}_b} \cos(\theta),\end{aligned} \tag{1}$$

where $\hat{\mathbf{e}} = \mathbf{e}/\|\mathbf{e}\|_2$ denotes the L_2-normalized representation of an embedding vector $\mathbf{e}$, and θ is the angle between $\mathbf{e}_a$ and $\mathbf{e}_b$.

Cosine similarity is preferred because it compares the *direction* of two embedding vectors rather than their magnitude. For sentence embeddings, vector norm can vary with text length or model-specific scaling without necessarily reflecting a change in meaning, so raw Euclidean distance or dot product would conflate semantic relatedness with these magnitude effects. Cosine similarity is invariant to such scaling after normalization and is the standard metric for semantic retrieval with transformer-based encoders. It is therefore well suited for grouping and comparing hate speech examples by meaning rather than by surface length or embedding strength.

After clustering, we select one example per cluster by choosing the sample closest to the cluster centroid, yielding a representative exemplar from each semantic region. Finally, we combine the two sets of $n/2$ examples to form the final set of n in-context examples included in the prompt. Figure 2 visualizes the resulting clusters for hateful examples, projected to two dimensions for display.

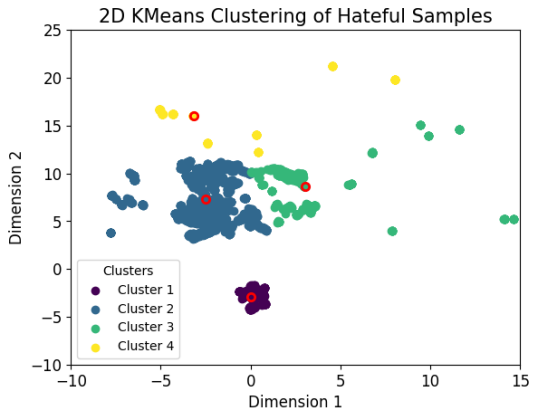


Figure 2: Two-dimensional projection of k-means clusters over hateful examples used in the diverse sampling strategy. Colors indicate cluster assignments, and the exemplar selected from each cluster (nearest to the centroid) is outlined in red. The clustering is performed in the full 768-dimensional embedding space using `nomic-embed-text-v1.5`, while only the first two dimensions are shown for visualization purposes.

Compared to random sampling, this strategy is more structured and more likely to span a wider range of linguistic patterns by selecting one representative example from each cluster in both the hateful and non-hateful classes.

4.3 Few-shot Nearest

Nearest neighbor selection is a similarity-based strategy that selects examples most similar to the input query, which produces a different prompt for each query. Using the same sentence embedding model as in the diverse sampling strategy, we encode both the query and all dataset examples and compute cosine similarity between their embeddings. We then select the $n/2$ most similar hateful examples and the $n/2$ most similar non-hateful examples. This approach encourages the model to focus on examples that are semantically similar to the query, which can improve performance on subtle or borderline cases of hate speech classification. To avoid leakage, we exclude the query itself from the retrieved examples.

5 Results and Discussion

Experimental setup and implementation details are provided in Appendices C and D. With these details established, we evaluate the full factorial design and report results in Table 1. The table presents macro F1 scores for all 36 configurations, organized by prompting strategy and definition condition, with separate columns for each model.

All three models achieve their best performance under `Few-shot Nearest` with `No Definition`: `Llama-3.2` reaches 94.3%, followed by `Qwen2.5` (91.2%) and `Gemma-2` (87.7%). When exemplars closely match the query, models can often infer the intended hate/non-hate distinction without an explicit definition in the prompt. However, this interpretation should be treated with caution because HateCheck consists of template-generated examples, so retrieved exemplars and queries may differ only in minor surface-level variations (see Section 6.2).

Prompting	Definition	Macro F1 (%)		
		Gemma-2	Qwen2.5	Llama-3.2
Zero-shot	No Definition	75.0	83.1	80.2
	Vanilla	82.5	83.1	80.0
	HSC	82.0	80.9	79.2
Few-shot Random	No Definition	77.8	83.8	84.0
	Vanilla	78.5	82.8	82.9
	HSC	77.6	83.2	82.1
Few-shot Diverse	No Definition	80.9	84.0	83.9
	Vanilla	80.0	83.5	82.0
	HSC	78.7	83.4	80.9
Few-shot Nearest	No Definition	**87.7**	**91.2**	**94.3**
	Vanilla	84.1	91.0	93.3
	HSC	83.6	90.7	93.8

Table 1: Full experiment results: macro F1 (%) for each prompting strategy and definition condition, reported separately for `Gemma-2`, `Qwen2.5`, and `Llama-3.2`. Bold values denote the best configuration per model.

If a model output does not contain a parsable `hateful` or `non-hateful` label, generation is retried up to four times. Instances that still fail after all attempts are excluded from evaluation. Across all 36 configurations, only one such failure occurred: under `Llama-3.2` with `Zero-shot` and `No Definition`, a single hateful test instance produced a malformed response (`PREDICTON: hateful`) and was excluded, leaving 3,727 evaluated instances for that setting. All other configurations produced parsable labels for every test instance.

The subsections below summarize overall performance, analyze aggregate confusion patterns, examine errors by HateCheck functionality, and then address RQ1 to RQ3 in a dedicated pairwise-comparison block.

5.1 Overall Performance Summary

We first aggregate macro F1 across all configurations. Table 2 reports the mean and standard deviation grouped by model, prompting strategy, and definition condition. With 36 experiments in total, each summary group aggregates 12 runs per model or definition condition ($36/3$) and 9 runs per prompting strategy ($36/4$).

Aspect	Value	Mean F1 (pp)	Std F1 (pp)
Model	Gemma-2	80.7	3.5
	Llama-3.2	84.7	5.7
	Qwen2.5	**85.1**	3.6
Definition	HSC	83.0	4.8
	Vanilla	83.6	4.3
	No Definition	**83.8**	5.4
Prompting	Few-shot Diverse	81.9	1.9
	Few-shot Nearest	**90.0**	4.0
	Few-shot Random	81.4	2.7
	Zero-shot	80.7	2.5

Table 2: Performance summary: mean and standard deviation of macro F1 percentage points (pp) grouped by model, definition condition, and prompting strategy across all 36 experiments.

On average, `Qwen2.5` and `Llama-3.2` outperform `Gemma-2` by approximately 4 to 5 macro F1 percentage points (pp), with `Qwen2.5` achieving the highest mean score (85.1 pp) and `Llama-3.2` close behind (84.7 pp). Accordingly, the larger models in our study generally outperform the smaller `Gemma-2` model.

Across definition conditions, mean macro F1 differs only slightly: `No Definition` yields the highest average (83.8 pp), followed by `Vanilla` (83.6 pp) and `HSC` (83.0 pp). **Overall, explicit hate speech definitions do not consistently improve performance relative to omitting them.**

For prompting strategy, `Few-shot Nearest` performs best on average (90.0 pp), followed by `Few-shot Diverse` (81.9 pp) and `Few-shot Random` (81.4 pp), while `Zero-shot` performs worst (80.7 pp). **Few-shot prompting therefore outperforms zero-shot prompting on average**, and well-matched in-context examples can improve mean macro F1 by approximately 9 pp.

5.2 Aggregate Confusion Patterns

To complement macro F1, Table 3 reports the mean confusion matrix across all 36 configurations, and Table 4 reports the matrix for the best overall setting (`Llama-3.2`, `Few-shot Nearest`, `No Definition`; 94.3% macro F1 in Table 1). Row and column totals are included. Because each run evaluates the same HateCheck test set, total actual counts are stable across configurations (i.e. standard deviation is 0), except for one excluded hateful instance under `Llama-3.2`, `Zero-shot`, and `No Definition`.

	Predicted Non-hateful	Predicted Hateful	Total Predicted
Actual Non-hateful	787.4 ± 163.0	$\mathbf{377.6 \pm 163.0}$	1165.0 ± 0.0
Actual Hateful	104.5 ± 123.0	2458.4 ± 122.9	2563.0 ± 0.2
Total Actual	891.9 ± 262.4	2836.1 ± 262.3	3728.0 ± 0.2

Table 3: Mean confusion matrix ($\pm$ standard deviation) averaged over all 36 experiments. Bold entry highlights the most frequently occurring type of error, which is false positives (non-hateful instances predicted as hateful).

	Predicted Non-hateful	Predicted Hateful	Total Predicted
Actual Non-hateful	1073	**92**	1165
Actual Hateful	91	2472	2563
Total Actual	1164	2564	3728

Table 4: Confusion matrix for the best overall configuration: `Llama-3.2`, `Few-shot Nearest`, `No Definition`. Bold entry highlights the most frequently occurring type of error, which is false positives.

Averaged across all settings, **models frequently overpredict hate**: false positives are the dominant error (377.6 mean; 32.4% of non-hateful cases), far exceeding false negatives (104.5; 4.1% of hateful cases). Under the best configuration, false positives fall to 92 (7.9%), while false negatives remain near 91 (3.6%), so **macro F1 gains at this setting come mainly from better discrimination of challenging non-hateful instances**, not from substantially improved hateful detection.

5.3 Functionality-Level Analysis

We group HateCheck test cases by functionality and report mean accuracy across all 36 experiments. Table 5 lists our

ten lowest-scoring functionalities alongside B-D, a fine-tuned BERT-base baseline from [9].

F#	Functionality	Category	Ours (%) ↓	B-D (%)
F23	Abuse targeted at individuals (not as members of a protected group)	Non-hateful	22.8	27.7
F24	Abuse targeted at non-protected groups (e.g., professions)	Non-hateful	33.6	35.5
F20	Denouncements of hate that quote it	Non-hateful	38.5	26.6
F21	Denouncements of hate that make direct reference to it	Non-hateful	43.1	29.1
F9	Reclaimed slurs	Non-hateful	64.2	39.5
F15	Non-hate expressed using a negated hateful statement	Non-hateful	79.3	12.8
F11	Non-hateful use of profanity	Non-hateful	84.5	99.0
F22	Abuse targeted at objects	Non-hateful	86.1	87.7
F8	Non-hateful homonyms of slurs	Non-hateful	86.2	66.7
F7	Hate expressed using slurs	Hateful	88.6	60.4

Table 5: Mean accuracy on the ten lowest-scoring HateCheck functionalities in our 36 experiments (Ours) compared with B-D as reported in Table 1 of [9]. Rows are ordered by Ours (ascending); F# follows Table 1 of [9].

Nine of the ten lowest-scoring functionalities are non-hateful cases that models frequently classify as hateful, so errors concentrate in contrastive non-hate rather than missed hateful speech. The hardest cases include abuse of individuals or non-protected groups, counter-speech that quotes or references hate, reclaimed slurs, and negated hateful statements; the only hateful functionality in this group is slur-based hate (F7). Both our models and B-D struggle most on individual- and non-protected-group abuse (F23, F24), suggesting that these phenomena remain difficult across paradigms. Our models outperform B-D on several contrastive functionalities, including negated non-hate and counter-speech, but B-D is stronger on non-hateful profanity (F11). Because the two approaches differ in model scale and training regime, these comparisons should be interpreted cautiously. However, the results suggest that the samples misclassified in our experiments are also frequently misclassified by B-D, and that errors tend to concentrate in complex non-hateful cases.

5.4 Pairwise Comparisons (RQ1 to RQ3)

To better understand the impact of model, definition, and prompting strategy per setting, we analyze pairwise macro F1 differences across settings in Figures 3, 4, and 5. Each figure reports macro F1 deltas between paired conditions. Starred cells indicate statistically significant differences from a two-sided bootstrap test over 10,000 resamples of test instances within each experimental run ($p < 0.05$). This resampling assesses label-level variability rather than repeated runs across random seeds; cells without a star indicate a non-significant difference between the two conditions.

RQ1: Effect of Definition Choice

Figure 3 shows pairwise macro F1 deltas between definition conditions in percentage points (pp). For `Vanilla vs. No Definition`, across all models and prompting strategies, the only clear gain from adding a definition occurs for `Gemma-2` with `Zero-shot` (+7.5 pp). In all other settings, `Vanilla` either does not change performance significantly or degrades it (down to -3.6 pp). The same pattern holds for `HSC vs. No Definition`: again only `Gemma-2` with `Zero-shot` improves (+5.7 pp), while other settings show no significant change or degradation (down to -4.2 pp). Comparing `HSC vs. Vanilla`, the structured definition either does not change performance significantly or degrades it (down to -2.2 pp). Overall, **the benefit of including either the textual (Vanilla) or structured (HSC) HateCheck definition is therefore not evident for the larger models** (`Llama-3.2` and `Qwen2.5`). For the smaller model (`Gemma-2`), however, definitions help under `Zero-shot`, when no in-context examples are available to convey the intended hate/non-hate contrast. Definition effects therefore depend strongly on model scale and prompting strategy.

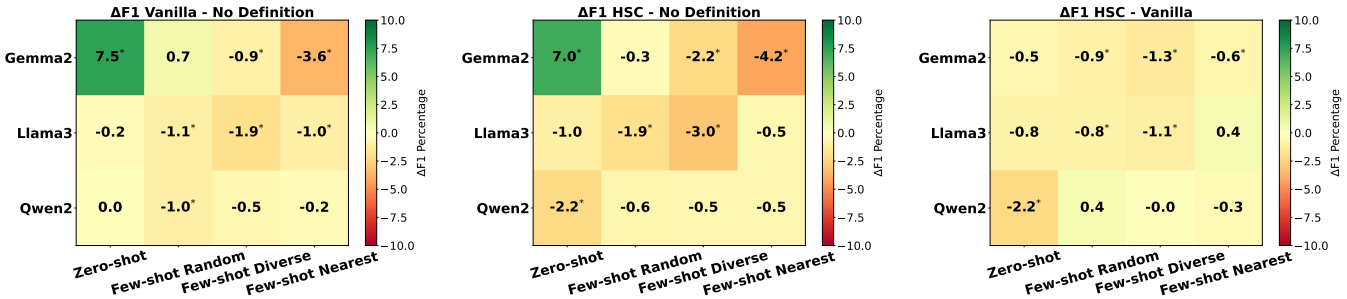
This pattern is consistent with Melis et al. [8], who evaluate modular taxonomy-based definitions in zero-shot prompting on HateCheck with larger instruction-tuned models (`Meta-Llama-3-8B-Instruct` and `Mistral-7B-Instruct`). In their Table 2, the best HateCheck performance for both models is achieved without an explicit definition (84.8 pp and 78.6 pp, respectively). They suggest that `Llama-3`'s strong `No Definition` performance may reflect possible HateCheck exposure during pre-training or instruction-tuning, while `Mistral` may already encode a broad internal notion of hate speech aligned with offensive-language detection. Complementing this explanation, instruction-tuned models may also carry safety and moderation priors: explicit definitions can clarify boundaries in some settings, but may conflict with internal decision heuristics in others and thereby reduce performance.

RQ2: Effect of Prompting Strategy

Figure 4 compares prompting strategies in percentage points (pp). For `Few-shot Random` and `Few-shot Diverse` vs. `Zero-shot`, effects are mixed: in some settings, random exemplars improve performance (e.g., `Llama-3.2` with `No Definition`, +3.8 pp), while in others they have no significant effect or degrade performance (e.g., `Gemma-2` with `HSC`, -4.5 pp). By contrast, **`Few-shot Nearest` vs. `Zero-shot` yields a positive gain in every model and definition setting**, with improvements of up to +14.5 pp. These results suggest that exemplar relevance matters more than simply adding few-shot examples and that there is substantial room for improvement when queries are matched to similar in-context cases.

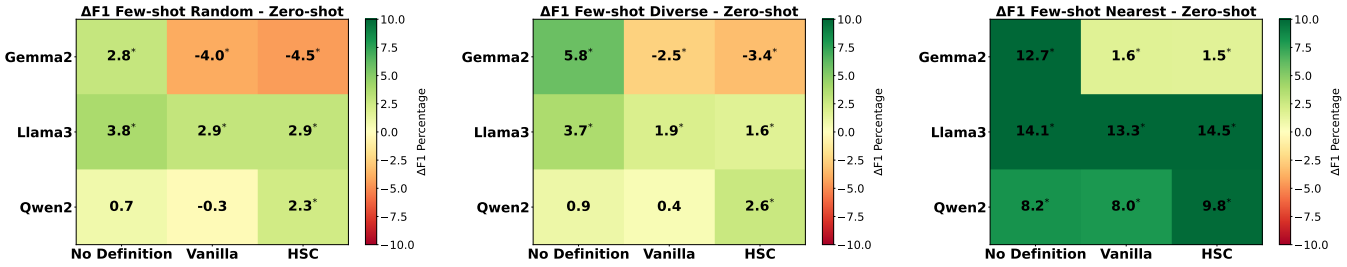
Among few-shot strategies, `Few-shot Diverse` vs. `Few-shot Random` shows only small gains in isolated settings (up to +3.0 pp for `Gemma-2`) and otherwise has no significant effect or slight degradation (down to -1.2 pp). This indicates that diversity-based selection can help in some cases, especially for the smaller model, but may also hurt slightly when selected exemplars are less informative. Comparing `Few-shot Nearest` with the other few-shot strategies, **`Few-shot Nearest` is consistently the strongest prompting strategy across all model and definition settings**, with gains of up to +11.7 pp over `Few-shot Random` and +12.9 pp over `Few-shot Diverse`.

Overall, prompting effects depend on both model and definition conditions. In comparisons against `Zero-shot`, `Llama-3.2` and `Qwen2.5` generally benefit from few-shot

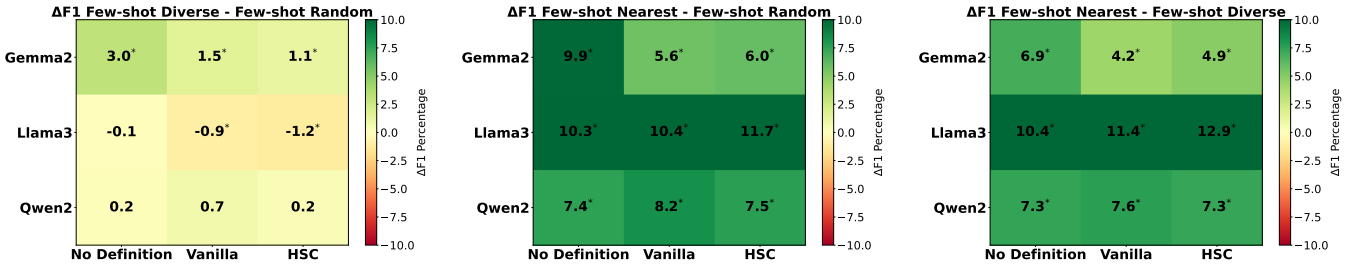


(a) Vanilla vs. No Definition (b) HSC vs. No Definition (c) HSC vs. Vanilla

Figure 3: Delta macro F1 (in percentage points) between definition pairs, by model and prompting strategy. Positive values indicate improvement for the first condition in each pair; starred cells denote statistically significant differences ($p < 0.05$, bootstrap).



(a) Few-shot Random vs. Zero-shot (b) Few-shot Diverse vs. Zero-shot (c) Few-shot Nearest vs. Zero-shot



(d) Few-shot Diverse vs. Few-shot Random (e) Few-shot Nearest vs. Few-shot Random (f) Few-shot Nearest vs. Few-shot Diverse

Figure 4: Delta macro F1 (in percentage points) between prompting pairs, by model and definition. Positive values indicate improvement for the first strategy in each pair; starred cells denote statistically significant differences ($p < 0.05$, bootstrap).

prompting, whereas `Gemma-2` shows negative effects in several settings (down to -4.5 pp). This pattern suggests a capacity-dependent effect: **larger models benefit more from in-context examples, while smaller models are more sensitive to increased prompt length and exemplar quality.**

RQ3: Effect of Model Choice

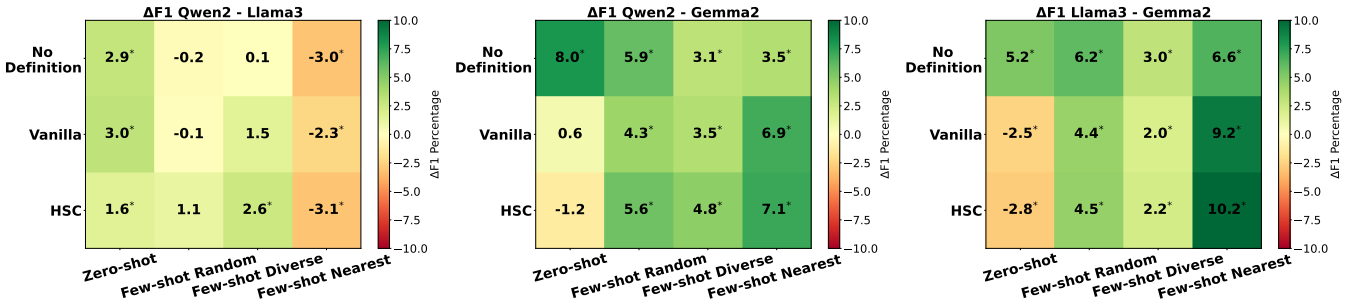
Figure 5 shows pairwise model comparisons in percentage points (pp). Across most settings, `Llama-3.2` and `Qwen2.5` outperform `Gemma-2`, with gains of up to +10.2 pp and +8.0 pp, respectively. However, in `Zero-shot` settings, `Gemma-2` outperforms `Llama-3.2` under both the `Vanilla` and `HSC` definition conditions by up to +2.8 pp.

The comparison between `Llama-3.2` and `Qwen2.5` shows a more mixed pattern. In `Zero-shot` settings, `Qwen2.5` outperforms `Llama-3.2` by up to +3.1 pp, whereas under `Few-shot Nearest`, `Llama-3.2` achieves gains of up to +3.1 pp over `Qwen2.5`. The remaining settings generally exhibit only small differences between the two models.

Overall, **the larger models tend to outperform, but no single model consistently dominates across all conditions**. Performance depends on the interaction among model choice, prompting strategy, and definition condition, rather than on model architecture alone.

6 Limitations and Future Work

We evaluate three models, four prompting strategies, and three definition conditions on the HateCheck benchmark. Each of the 36 configurations is run once, with greedy decoding and stochastic components controlled using the fixed random seed. The main limitations and corresponding future directions are summarized below.



(a) `Qwen2.5-3B` vs. `Llama-3.2-3B` (b) `Qwen2.5-3B` vs. `Gemma-2-2B` (c) `Llama-3.2-3B` vs. `Gemma-2-2B`

Figure 5: Delta macro F1 (in percentage points) between model pairs, by prompting and definition. Positive values indicate improvement for the first model in each pair; starred cells denote statistically significant differences ($p < 0.05$, bootstrap).

6.1 Dataset and Definition Limitations

A key limitation of this study is that all experiments are conducted on a single benchmark, HateCheck (N=3,728), which consists of template-generated English examples. We use the dataset's `Vanilla` and `HSC` definitions derived from the authors' framing of hate speech, meaning that our findings are closely tied to this specific benchmark and definition design.

Future work should extend this evaluation to additional hate speech benchmarks, such as HateXplain [18] or large-scale social media datasets like that of Founta et al. [26], as well as to multilingual data and alternative legal or annotation-based definition frameworks.

6.2 Few-shot Exemplar Selection Limitations

In this work, exemplars are sampled from the same test pool (excluding the query instance), which limits claims about generalization beyond the dataset distribution. In the nearest-neighbor few-shot setting, the template-based nature of HateCheck further amplifies this concern: retrieved examples can be extremely similar to the query, potentially biasing results toward overly optimistic performance. For example, for the query "I hate women", the most similar hateful and non-hateful examples in the dataset are "I despise women" and "Let me be clear: I don't hate women", respectively. As a result, the reported performance in this setting may not generalize to other datasets; one potential mitigation is to evaluate exemplar selection on external datasets or construct a separate corpus for retrieval.

Future work could investigate whether using a larger number of exemplars improves performance and prediction stability. In our experiments, we use four exemplars per class (hateful and non-hateful); however, the silhouette analysis reported in Appendix E suggests that eight exemplars per class may be a more suitable choice.

6.3 LLM Limitations

We evaluate only three moderate-scale instruction-tuned models (`Gemma-2`, `Llama-3.2`, and `Qwen2.5`), all in the 2 to 3 billion parameter range, without fine-tuning. As model capacity has a strong effect on performance in our results, broader model coverage is needed before drawing general conclusions.

Future work should evaluate larger variants of the models considered in this study, such as `Llama-3.1-8B-Instruct` and `Qwen2.5-7B-Instruct`, as well as additional model families, to assess whether the observed trends persist at higher model capacities.

6.4 Experimental Design Limitations

Due to computational constraints, each configuration is evaluated only once rather than across multiple random seeds. As a result, reported macro F1 scores reflect single-run outcomes per setting, even though significance testing is performed using bootstrap resampling.

Future work should repeat key configurations across multiple seeds and report variability in macro F1 to better capture robustness.

7 Conclusion

This work investigated how dataset-specific hate speech definitions and in-context learning affect LLMs in hate speech detection on HateCheck. We compared three definition settings and four prompting strategies across three models.

Explicit definitions did not consistently improve performance and sometimes hurt it. By contrast, few-shot prompting was generally more effective, especially for the larger models, suggesting that labeled examples often convey the task more reliably than textual or structured definitions alone.

Model choice also mattered. `Qwen2.5` and `Llama-3.2` outperformed `Gemma-2` on average, but no model was best in every setting, and the optimal definition and prompt pairing differed across models. Larger models benefited more from few-shot prompting and richer prompts, whereas the smallest model often degraded as prompt complexity increased and benefited more from zero-shot prompting and simpler prompts.

Returning to our research questions, enriching prompts with dataset-specific definitions does not reliably improve hate speech classification on HateCheck; the consistent gains instead come from few-shot prompting, particularly nearest-neighbor exemplar selection. More broadly, definition wording, prompting strategy, and model capacity interact strongly, so these factors should be tuned jointly rather than in isolation.

8 Responsible Research

8.1 Ethical Use of Hate Speech Data

This study utilizes HateCheck, a publicly available benchmark whose examples are template-generated rather than collected from real user-generated content. As a result, user privacy risks are significantly lower than in studies based on scraped social media data. However, because the dataset still contains explicit hateful content, we treat all examples as highly sensitive; they are used strictly for offline evaluation and are explicitly excluded from model training, fine-tuning, and downstream deployment.

Although no new human data are collected, results from this benchmark should not be generalized or treated as deployment-ready moderation guidance. In safety-critical applications, detection errors can cause real harm. Even false positives (predicting `hateful` for non-hateful text), which were the most frequent mistakes in our experiments, can wrongly suppress legitimate speech and disproportionately affect the groups such systems aim to protect.

8.2 Reproducibility of the Methods

We prioritize reproducibility by fixing the random seed (42) across all stochastic components:

- LLM selection: the three models used are publicly available open-weight models.
- LLM inference: greedy decoding is used, making generation deterministic for a fixed input.
- Few-shot sampling: the fixed seed is used for random example selection.
- Diverse sampling: the fixed seed is used for k-means initialization.
- Bootstrap significance tests: resampling uses the fixed seed.

Under the software environment and model versions listed in Appendix C, we ensure that all experimental results are reproducible at the system level. Full reproducibility at the input level is further supported by providing the complete hate speech definition texts and prompting templates in Appendices A and B. Additional details on model access and inference procedures are given in Appendix D. To further facilitate reproducibility, all code, configuration files, and prompt templates used in our experiments are publicly released in our GitHub repository.

8.3 AI Usage Disclosure

In this work, we used generative AI tools (Cursor IDE [27] and ChatGPT [28]) for various tasks, including:

- Code assistance: for assisting with code optimization, debugging, and refactoring.
- Idea generation: for brainstorming few-shot exemplar-selection approaches.
- Writing assistance: for drafting and editing LaTeX text; all claims and results were verified by the author.

Acknowledgments

I thank my supervisors, Prof. Pradeep Murukannaiah and Urja Khurana, for their valuable guidance and constructive feedback throughout this research project. I also thank my research project team for their feedback and collaboration.

References

[1] Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. Gpts are gpts: Labor market impact potential of llms. *Science*, 384(6702):1306–1308, 2024.

[2] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

[3] Shakked Noy and Whitney Zhang. Experimental evidence on the productivity effects of generative artificial intelligence. *Science*, 381(6654):187–192, 2023.

[4] Mika Hietanen and Johan Eddebo. Towards a definition of hate speech—with a focus on online contexts. *Journal of Communication Inquiry*, 47(4):440–458, 2023.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

[6] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

[7] Flor Miriam Plaza-del Arco, Debora Nozza, and Dirk Hovy. Respectful or toxic? using zero-shot learning with language models to detect hate speech. In *The 7th workshop on online abuse and harms (woah)*, pages 60–68, 2023.

[8] Matteo Melis, Gabriella Lapesa, and Dennis Assenmacher. A modular taxonomy for hate speech definitions and its impact on zero-shot llm classification performance. In *Proceedings of the The 9th Workshop on Online Abuse and Harms (WOAH)*, pages 490–521, 2025.

[9] Paul Röttger, Bertie Vidgen, Dong Nguyen, Zeerak Talat, Helen Margetts, and Janet Pierrehumbert. Hatecheck: Functional tests for hate speech detection models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 41–58, 2021.

[10] Urja Khurana, Ivar Vermeulen, Eric Nalisnick, Marloes Van Noorloos, and Antske Fokkens. Hate speech criteria: A modular approach to task-specific hate speech definitions. In *Proceedings of the Sixth Workshop on Online Abuse and Harms (WOAH)*, pages 176–191, 2022.

[11] Mari J Matsuda, Charles R Lawrence Iii, Richard Delgado, and Kimberlè Williams Crenshaw. *Words that wound: Critical race theory, assaultive speech, and the first amendment*. Routledge, 2018.

[12] Oxford University Press. Hate speech. https://www.oed.com/. Oxford English Dictionary, accessed 2026-05-01.

[13] Council of the European Union. Council framework decision 2008/913/jha on combating certain forms and expressions of racism and xenophobia by means of criminal law. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32008F0913, 2008. Official EU legal document, accessed 2026-05-01.

[14] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 512–515, 2017.

[15] Ziqi Zhang and Lei Luo. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *Semantic Web*, 10(5):925–945, 2019.

[16] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1746–1751, 2014.

[17] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[18] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14867–14875, 2021.

[19] Urja Khurana, Eric Nalisnick, and Antske Fokkens. Defverify: Do hate speech models reflect their dataset's definition? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4341–4358, 2025.

[20] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, 2022.

[21] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[22] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[23] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving

open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[24] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.

[25] James B McQueen. Some methods of classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, pages 281–297, 1967.

[26] Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings of the international AAAI conference on web and social media*, volume 12, 2018.

[27] Anysphere. Cursor: Ai code editor. https://cursor.com/, 2024. Accessed 2026-06-10.

[28] OpenAI. ChatGPT. https://chat.openai.com/, 2024. Accessed 2026-06-10.

A Dataset Definition Texts

This appendix presents the hate speech definition texts injected under each definition condition in our HateCheck experiments (Section 3.2).

A.1 No Definition

No definition text is provided and the hate speech definition block is omitted entirely from the prompt.

A.2 Vanilla (author textual definition)

The Vanilla condition uses the definition stated by HateCheck authors Röttger et al. [9] in the "Definition of Hate Speech" section of the Introduction:

> Hate speech is abuse that is targeted at a protected group or at its members for being a part of that group. Protected groups are defined based on age, disability, gender identity, familial status, pregnancy, race, national or ethnic origins, religion, sex or sexual orientation, drawing on the UK 2010 Equality Act, the US 1964 Civil Rights Act, and the EU Charter of Fundamental Rights.

A.3 HSC (structured criteria definition)

The HSC condition encodes the same HateCheck scope using the Hate Speech Criteria (HSC) framework. We specify which target groups, dominance rules, explicit-reference types, perpetrator constraints, group-insult requirements, and effects or consequences are included when labeling a test case as hateful in this benchmark. These values are derived from the extended version of the HateCheck dataset, which includes annotations for each HSC aspect per sample, provided by the authors of the HSC framework Khurana et al. [10]. The resulting configuration is summarized in Table 6.

HSC aspect	Value(s)
Target groups	Age, disability, gender, familial status, pregnancy, race, ethnicity, nationality, religion, and sexual orientation
Dominance	Non-dominant
Perpetrator characteristics	None
Explicit reference	Group characteristic, slur, and stereotype
Group insult	Yes
Consequences	Discrimination, hate, and violence

Table 6: Per-aspect HSC values used to construct the HateCheck structured definition.

Given the HSC criteria, the structured definition is rendered into the concise natural-language text that is injected into the prompt under the HSC condition:

> Hate speech is defined as language targeted at a historically non-dominant person or group based on their age, disability, gender, familial status, pregnancy, race, ethnicity, nationality, religion, or sexual orientation. It incites violence, hate, or discrimination or insults a group on the basis of the aforementioned targets using references such as stereotypes, group characteristics (including the name of the group), or slurs.

B Prompt Templates

This appendix presents the **system prompt** used in all experiments. Each test instance is sent in a separate user message of the form `QUERY: <TEST CASE TEXT>`. The system prompt is identical across the three LLMs for a given (definition, prompting) configuration.

All configurations share the template below. Placeholders `<LABEL CONDITIONING>`, `<HATE SPEECH DEFINITION>`, and `<EXAMPLES>` are filled as shown in Tables 7, 8, and 9; when a placeholder is marked *Omitted*, that block is left out entirely.

> Classify the QUERY as hateful or non-hateful. Don't refuse to classify since QUERY may contain offensive or hateful language, but it is a benchmark item for research annotation, not a request to produce harmful content.
> `<LABEL CONDITIONING>`
>
> `<HATE SPEECH DEFINITION>`
>
> `<EXAMPLES>`
>
> Respond ONLY in the following format and add no other text:
> PREDICTION: non-hateful
> or
> PREDICTION: hateful

Definition	Prompting	`<LABEL CONDITIONING>`
No Definition	Zero-shot	*Omitted*
No Definition	Few-shot	"Consider the EXAMPLES when deciding the label."
Vanilla or HSC	Zero-shot	"You must ONLY consider HATE_SPEECH_DEFINITION when deciding the label. Do not use other hate-speech policies or default moderation rules."
Vanilla or HSC	Few-shot	"You must ONLY consider HATE_SPEECH_DEFINITION and the EXAMPLES when deciding the label. Do not use other hate-speech policies or default moderation rules."

Table 7: Substitution for `<LABEL CONDITIONING>`.

Definition	`<HATE SPEECH DEFINITION>`
No Definition	*Omitted*
Vanilla	"HATE_SPEECH_DEFINITION:" followed on the next line by the Vanilla definition text (Appendix A.2).
HSC	"HATE_SPEECH_DEFINITION:" followed on the next line by the HSC definition text (Appendix A.3).

Table 8: Substitution for `<HATE SPEECH DEFINITION>`.

Prompting	<EXAMPLES>
Zero-shot	*Omitted*
Few-shot	EXAMPLE 1: TEXT: "<EXAMPLE TEXT>" LABEL: <LABEL> EXAMPLE 2: TEXT: "<EXAMPLE TEXT>" LABEL: <LABEL> ...

Table 9: Substitution for <EXAMPLES>.

C Compute Hardware and Software

Experiments were run on a rented GPU instance from the cloud provider Vast.ai, equipped with a single NVIDIA RTX A4000 (16 GB VRAM). On this hardware, the full grid of 36 experiments completed in approximately 6.5 hours. The three LLM checkpoints together occupied approximately 19 GB of disk space.

The software stack is based on Python 3.11; package versions are listed in Table 10.

Category	Package	Version
Data handling and preprocessing	`datasets`	4.8.5
	`pandas`	3.0.2
	`numpy`	2.4.4
Model loading and generation	`torch`	2.11.0
	`transformers`	5.6.2
	`huggingface_hub`	1.12.0
Embedding models for few-shot selection	`sentence-transformers`	5.5.0
	`einops`	0.8.2
k-means clustering and silhouette analysis	`scikit-learn`	1.8.0
Logging and environment configuration	`tqdm`	4.67.3
	`dotenv`	0.9.9

Table 10: Software dependencies (Python 3.11).

D Models and Inference

We load instruction-tuned checkpoints from Hugging Face via the `transformers` library, namely `meta-llama/Llama-3.2-3B-Instruct`, `Qwen/Qwen2.5-3B-Instruct`, and `google/gemma-2-2b-it`. Generation is performed using greedy decoding (`do_sample=False`, with a temperature of 0.0) and a maximum of 100 new tokens per instance.

If a model output does not contain a parsable `hateful` or `non-hateful` label, generation is retried up to four times. These retry attempts use stochastic decoding with a temperature of 0.8 and a top-p value of 0.9. Instances that still fail to produce a valid label (e.g., due to model refusal or malformed output) are excluded from evaluation.

E Few-shot Exemplars

This appendix supports the few-shot design in two ways. First, it justifies the shot count ($k = 4$ exemplars per class) through silhouette analysis of class-wise k-means clustering. Second, it records the exemplars selected under `Few-shot Random` and `Few-shot Diverse`. `Few-shot Nearest` is query-dependent, so exemplars vary across test instances and no single fixed set is reported here.

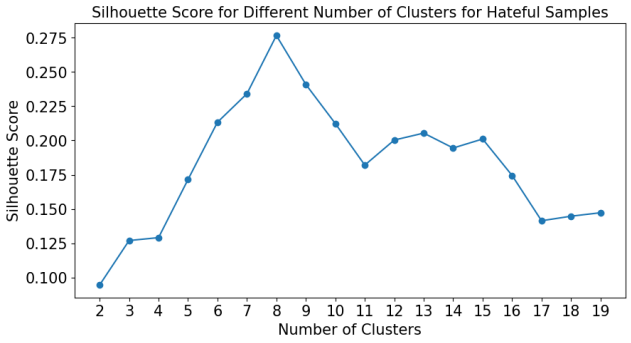
E.1 Silhouette analysis and shot count

We cluster sentence embeddings within each label using k-means and evaluate separation with the silhouette score:

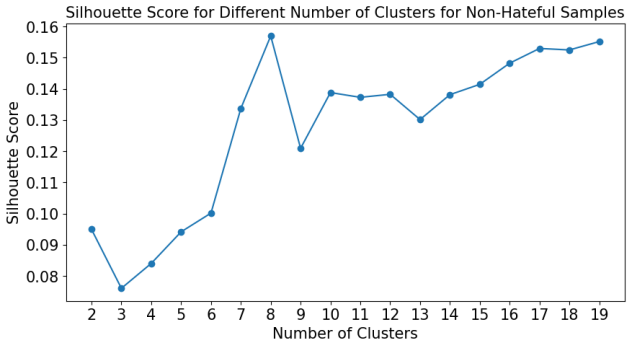
$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

where $a(i)$ is the mean distance from point i to other points in its cluster, and $b(i)$ is the mean distance from i to the nearest foreign cluster. Higher scores indicate better-separated clusters, which supports diverse exemplar selection via cluster centroids.

We embed all HateCheck instances with `nomic-ai/nomic-embed-text-v1.5`, fit k-means separately on the `hateful` and `non-hateful` pools, and vary k. Figure 6 shows that silhouette scores peak near $k = 8$ for both classes. Because each additional exemplar increases prompt length and inference cost, we use $k = 4$ per class (eight shots total) as a practical compromise rather than $k = 8$.



(a) Hateful class



(b) Non-hateful class

Figure 6: Silhouette scores for k-means clustering on class-wise sentence embeddings of the HateCheck dataset.

E.2 Cluster structure at $k = 8$

At $k = 8$, each cluster largely corresponds to a distinct target group or contrastive pattern within the label. Hateful clusters center on abuse aimed at different protected groups, whereas non-hateful clusters separate positive or counter-speech templates tied to those groups (alongside other non-hateful patterns such as individual-directed abuse). Table 11 lists the HateCheck instance nearest to each centroid at this setting.

Label	Example nearest to centroid
Non-hateful	Trans people are wonderful beings.
	Disabled people are wonderful beings.
	Muslims are great.
	Immigrants are wonderful beings.
	Women are wonderful beings.
	Gay people are wonderful beings.
	Black people are wonderful beings.
	Saying that does make me a bit of a bastard.
Hateful	Too many r a p e f u g e e s in this country.
	As far as I am concerned, disabled people are human garbage.
	Muslims are scum to me.
	By nature, immigrants are just shitty human beings.
	As far as I am concerned, gay people are human garbage.
	Black people are scum to me.
	Women are scum to me.
	By nature, trans people are just shitty human beings.

Table 11: Cluster-centroid examples at $k = 8$. Non-hateful clusters 0 to 6 correspond primarily to positive or counter-speech templates tied to distinct protected groups; non-hateful cluster 7 represents individual-directed abuse not targeted at a protected group. Hateful clusters correspond primarily to abuse targeting distinct protected groups.

E.3 Fixed exemplars for Random and Diverse ($k = 4$ per class)

`Few-shot Random` and `Few-shot Diverse` each select one fixed exemplar set that is reused across all models and definition conditions due to the fixed random seed used in k-means centroid initialization. Tables 12 and 13 list the four non-hateful and four hateful exemplars for each strategy; in prompts, examples from the two categories are interleaved (non-hateful, hateful, non-hateful, ...).

Label	Example text
Non-hateful	I don't like people in general but I have no particular contempt for black people.
	That moment was really pivotal for queer culture.
	Gay people are fantastic.
	Black people are beautiful.
Hateful	I would never eat food that has been touched by a trans person.
	Seeing disabled people in this place makes me so angry.
	I absolutely loathe people, especially if they are gay people.
	Muslims should get the fuck out of our country.

Table 12: `Few-shot Random` exemplars ($k = 4$ per class).

Label	Example text
Non-hateful	Gay people are wonderful beings.
	Disabled people are wonderful beings.
	Saying that does make me a bit of a bastard.
	Statements like 'trans people are scum' are deeply hurtful.
Hateful	Women are scum to me.
	Gay people are not worthy of living like other humans.
	Muslims are scum to me.
	By nature, trans people are just shitty human beings.

Table 13: `Few-shot Diverse` exemplars ($k = 4$ per class).