Physical Education, Health and Social Sciences

https://journal-of-social-education.org E-ISSN: 2958-5996

P-ISSN: 2958-5988

Cyberbullying Detection in Social Media Text using Transformer based Learning Models

Zahid Mehmood [1], Ali Saeed [2],

[1] Department of Applied Computing Technologies, UCP, Lahore, Pakistan. Email: L1F23MSDS0001@ucp.edu.pk

[2] Department of Software Engineering, FOIT, UCP, Lahore, Pakistan Email: Ali.saeed@ucp.edu.pk

DOI: https://doi.org/10.63163/jpehss.v3i4.970

Abstract

Cyberbullying on social media isn't a monolithic problem; it strikes people from all sides, including age, gender, ethnicity, and religion. As a result, moderation workflows clearly need fine-grained detection. Using the cyberbullying_tweets dataset (47,692 samples in stratified 70/15/15 splits), this paper benchmarks transformer-based learning models in the space of 6-class cyberbullying detection. The comparison examines two different paths. There's supervised fine-tuning via a QLoRA-style recipe with LoRA adapters and 4-bit NF4 quantization. Second is few-shot in-context learning based on chain-of-thought prompting and 12 examples per class (72 overall). This was done in eight instruction-tuned LLMs, including Qwen2.5 7B/8B, DeepSeek 7B/8B, Llama-3 7B/8B, Mistral-7B, and Mixtral-8x7B. Macro-F1 scores under supervised fine-tuning ranged from 0.809 to 0.857, which are identical to the accuracy range. The excellent model Mixtral-8x7B-Instruct reached macro-F1 0.857, with MCC 0.808, AUPRC 0.911, and a small calibration error (ECE 0.021). Few-shot prompting wasn't far behind, with macro-F1 between 0.770 and 0.820. The difference is usually only 0.03-0.04 compared to fine-tuning, but it significantly reduces training compute. In general, these results assess the trade-off between cost and performance and establish a reproducible baseline for transformer-based moderation.

Keywords: Cyberbullying detection, transformers, large language models, parameter-efficient fine-tuning, LoRA, QLORA, few-shot learning, chain-of-thought, calibration.

Introduction

The challenge to maintain social media safety, meanwhile, is difficult, a constant slog, with cyberbullying and abusive language that ranges from random harassment to targeted identity attacks. Most moderation pipelines fall back on binary toxicity detection, but all too often the solutions for interventions that are actually implemented require fine-grained categorization, because that's the only approach to auditing, policy enforcement, or responses proper. This research supports the observation that reveals the abusive language is context-bound which is why the model reliability suffers from factors. The factors such as domain shift, ambiguity or simply noise in the annotations [25-29]. And, shared tasks in hate speech detection have been obvious: So for that reason the transparent evaluation protocols and powerful models are required. [23,24,30].

Fine-grained detection is not all about simply flagging the harmful messages. It is also about separating out certain kinds (like age, religion, ethnicity, gender-based abuse) that could resemble each other linguistically. But also imply very different intentions and impacts. Early researches had focused on feature engineering and task-specific architectures. Those early engineering had showed good results in controlled setups [1]. Subsequent work embraced training strategies that tuned for affective signals. It also boosts detection for such things as defamation and harassment

[2]. However now, transformer-based models have fully revolutionized natural language processing. Mainly due to their remarkable empirical performance in text classification and transfer learning [4].

Transformer moderation systems is deployed along two practical lines in this paper. The first is supervised fine-tuning of large language models (LLMs)-though task-specific adaptation which increases accuracy. But it consumes a lot of computation power. Parameter efficient methods, including LoRA was used in it. That reduces training costs by updating low-rank adapters instead of having every model weight to be updated [10]. It also does quantization-aware recipes such as QLORA, which allow you to embed bigger backbones in tight GPU memory budgets [11]. Instead, the second direction? Few-shot in-context learning. In this way the models do not require gradient updates. But only requires the prompt demonstrations. A style that has been heavily adopted by larger models [16]. Chain-of-thought prompting adds structured reasoning, which improves reliability on complex decisions [15]. But with respect to cyberbullying detection at the granular level. We do not yet have a controlled comparison of these two strategies under consistent splits and constraints to fully inform system design.

This paper presents a comparative evaluation of transformer-based models for 6-class cyberbullying detection, centering on instruction-tuned LLMs in two settings: parameter-efficient supervised fine-tuning and few-shot inference with chain-of-thought. We emphasize calibration and operational metrics alongside standard classification performance because real deployments need well-calibrated confidence for thresholding, escalation, and human-in-the-loop review [12].

Contributions

The core outcomes of this work are listed below:

• A controlled benchmark for 6-class cyberbullying detection, using eight instruction-tuned transformer backbones under uniform stratified splits and evaluation procedures.

• A systematic comparison between parameter-efficient supervised fine-tuning (using LORA adapters and 4-bit quantization) and few-shot in-context inference with chain-of-thought prompting.

• Comprehensive reporting of reliability and effectiveness measures, including AUPRC [14], macro-F1, MCC [13], and calibration error (ECE) [12], along with operational data like peak GPU memory and training/inference time.

• A reproducible experimental setup, which supports fair comparisons and subsequent extensions in fine-grained cyberbullying moderation research.

Paper Organization

Section 2 reviews existing works on abusive language and cyberbullying detection focusing on the fine-grained taxonomies, efficient adaptation methods, and transformer-based models. Section 3 provides a description of the dataset, preprocessing steps, as well as stratified experimental splits. The modeling methodology, including few-shot in-context inference and supervised fine-tuning with parameter-efficient adaptation is presented in Section 4. The evaluation protocol, including calibration analysis and classification metrics, is defined in Section 5. The results and discussion of experimental data is included in Section 6. Last but not least, Section 7. It concludes this work and suggests future directions of research.

Literature Review

Abusive Language and Cyberbullying Detection

There has been a lot of academic attention on the automated discovery of harmful content on social media. It is often couched as comparable tasks. Like, detecting hate speech, pernicious content or offensive language. Significant, longitudinal studies show persistent challenges. The

challenges such as, it can be difficult to label stuff unambiguously. At times, targeted abuse and innocuous profanity resemble each other. The system sensitivity to shifts in platform or domain is a genuine problem [25-27]. Moreover, surveys and detailed analyses repeatedly highlight the extent to which annotation policies, dataset quality, and accidental bias severely distort model response. As well as the performance, thus highlighting the need for transparent benchmark comparison [28, 29]. Sharing tasks and benchmarks among the community has pushed progress by enabling joint consensus. Which enables for standardised evaluation approaches and label sets (and particularly for tracks around offensive content. Such as, those that show hate [23, 24]). At the same time, dedicated toxicity resources, such as the Jigsaw benchmarks. They have become very common to critique and program moderation system [30].

Fine-Grained Cyberbullying Taxonomies

Unlike simple toxic/non-toxic classification, fine-grained cyberbullying detection aims to identify the exact target or specific kind of abuse; this is more beneficial for support and moderation workflows. That said, these fine-grained scenarios have not been studied enough in prior research. Why? It is challenging to obtain high-quality, balanced datasets, and it is difficult to differentiate between abuse categories that are very close semantically. SOSNet is a typical work in this regard; it proposed a graph-based approach for multi-class detection of cyberbullying among victim-target categories such as age, gender, ethnicity, religion, and "other" [1]. This technique builds a similarity graph for tweets based on embedding closeness, using a graph convolutional network to pass signals between related posts. It employs data expansion approaches as well to deal with class imbalance [1]. Efforts like this demonstrate that fine-grained categorization can be achieved, but it remains clearly extremely sensitive to how the dataset is composed, where the class boundaries lie, and what models are used.

Transformer-Based Text Representation Learning

Because they are able to extract transferable representations from large pretraining steps, transformer architectures have gradually become the leading approach for NLP modeling [4]. Pretrained transformer encoders such as BERT and models that came after it (ROBERTa, DeBERTa, etc.)-generally outperformed previous recurrent neural and feature-based approaches on a wide variety of text classification problems [19-21]. As for cases of multi-lingual transfer or cross-lingual transfer, models like XLM-R enable relatively stable movement over different domains and languages [22]. These recent developments pave the way for constructing cyberbullying detection systems requiring fine-grained semantic understanding and contextual reading.

Efficient Adaptation of Large Language Models

Supervised fine-tuning certainly makes tasks perform better although training an entire large model is costly. Parameter-efficient fine-tuning (PEFT) techniques reduce computation requirements by only updating small adapter parameters rather than all the model weights. LORA, for example, embeds low-rank adapters into the transformer layers to enable efficient changes, whilst the backbone remains frozen [10]. QLORA goes further: pairing 4-bit quantization with those LoRA-based updates reduces the GPU memory consumption significantly. This enables bigger models to be trained even with low-availability hardware [11]. These methods work especially very good and excellent on cyberbullying detection. Where training often has to be repeated for other domains, languages, or label types.

Prompting, Few-Shot Learning, and Chain-of-Thought

Talking about llms large language models, a new sort of approach began to emerge. Few-shot in-context learning. In this method, it asks the model to do a function rather than using gradient updates. It does by incorporating demonstrations right into the prompt [16]. Instruction tuning and alignment strategies improve the model's compliance. It improves with the task instructions and the resulting output formats [17, 18]. Chain-of-thought prompting has also been useful, encouraging the model to display its intermediate steps. This reasoning process often boosts performance on tasks with subtle decision points. Although, the benefit can depend on how the prompt is tailored and what output constraints are utilized [15]. In recent work on cyberbullying, researchers have also investigated multi-class problems and low-resource environments. These studies suggest that simple transformer models may be ineffective on subtle or indirect incidents. So methods that introduce more signals like emotion into the training are relevant [2].

Reliability, Calibration, and the Evaluation Measures

For moderation tasks, accuracy alone does not tells the whole story. That's because to set thresholds, regulate escalations, and determine. When it is appropriate for a human to review content, the reliability of a model's confidence scores is critical. Since modern neural models are frequently miscalibrated, reporting metrics that integrate this calibration is increasingly significant [12]. In addition to the macro-averaged F1 score, the Matthews Correlation Coefficient (MCC) is quite useful in multi-class conditions, reflecting the overall quality of performance across all categories [13]. In addition, precision-recall analysis and AUPRC can provide valuable insights into performance trade-offs in an imbalanced or decision-specific setting [14]. Collectively, these metrics provide a deployment-focused assessment of transformer-based cyberbullying detection systems.

Dataset and Preprocessing

Dataset Source and Task Definition

The experiments work with the cyberbullying tweets dataset [3], which is publicly accessible and provides a database of social media posts (often expressed in tweet text) for detailed detection of cyberbullying. The task is formulated as single-label multi-class classification: given a tweet, you will try to predict exactly one cyberbullying category from a fixed label set.

Label Space

The dataset provides six mutually exclusive classes:

- **not_cyberbullying**: non-bullying or neutral content,
- **other_cyberbullying**: cyberbullying content that is not explicitly attributed to a protected/target category below,
- **age**: age-targeted cyberbullying,
- **ethnicity**: ethnicity-targeted cyberbullying,
- **gender**: gender-targeted cyberbullying,
- **religion**: religion-targeted cyberbullying.

This label design supports moderation settings where the type of abuse is required in addition to the presence of abuse.

Class Distribution

The dataset has 47,692 labeled samples. Across the six categories, the distribution is almost perfectly balanced-see Table 1 for the sample counts.

Train-Validation-Test Splits

In this paper, a stratified approach is used to maintain the same class ratios over each partition through a 70/15/15 split. This configuration yields approximately 33,384 training samples, with 7,154 samples. They are allocated to both validation and testing. This stratification helps to ensure fair comparisons between modeling techniques. The techniques such as supervised fine-tuning vs few-shot inference. Along with the methodologies that are evaluated on the same held out test data.

Table 1: Class distribution of the cyberbullying_tweets dataset (47,692 samples).

Class	**Count**	**Share (%)**
not_cyberbullying	7,945	16.66
other_cyberbullying	7,823	16.40
age	7,992	16.76
ethnicity	7,961	16.69
gender	7,973	16.72
religion	7,998	16.77
Total	**47,692**	**100.00**

Text Preprocessing

Preprocessing is purposefully light. It aims to cut through noise but without losing the lexical. Also the semantic signals that help catch abuse. The pipeline includes:

• Removal and normalization of whitespace URLs. • Basic form of normalization on punctuation and repeated characters. • Taking care of common social-media artifacts (e.g., tokenization formatting). • Deduplication of look alike tweet texts, that are prior to splitting which minimizes the train-test leakage.

No label aware transformation techniques are used at any of the stages. Every model is trained and tested on the identical processed text so that the results stay comparable.

Ethical Considerations

Because of its nature, this dataset holds content that is offensive and harmful. Managing and reporting these samples follows a minimization rule: instances aren't reproduced unless it's genuinely necessary to explain the methodology. When examples are shown, they're sanitized-meaning slurs and identifiers get masked. In this paper, the goal is to back up safer moderation workflows, not to give harmful language a bigger platform.

Methodology

Overview

Two transformer-based techniques for fine-grained cyberbullying detection are evaluated with identical data splits (Section 3) in this paper. As shown in Fig.1 and Fig. 2, the first method uses instruction-tuned large language models with parameter-efficient supervised fine-tuning. The second method considers few-shot in-context inference, which is independent of gradient updates. In order to keep it fair, the two approaches use the same preprocessing, stratified splits, and test data as before.

Model Backbones

Eight distinct instruction-tuned transformer backbones are employed in this work, covering several current model groups: Qwen2.5 (7B, 8B), DeepSeek (7B, 8B), Llama 3 (7B, 8B), Mistral-7B, and Mixtral-8x7B. These choices span various designs and training methods. They provide a panoramic view of the transformer model behaviour towards the detection of bullying. Details of these model families are in the original reports [31-34].

Supervised Fine-Tuning (SFT) with Quantized LORA Adapters

Efficient Adaptation

Supervised tuning is achieved using Low-Rank Adaptation (LoRA). This approach embeds small, trainable matrices in particular transformer components, and keeps the main weights frozen [10]. Models are loaded with 4-bit quantization (NF4) for memory saving. They are trained on a QLORA approach [11]. It is possible to fine-tune 7B-8B models on a single consumer-grade GPU without significantly dropping the prediction quality, due to this configuration.

Training Goal and Format

The job is executed as a single-label, multi-class classification using instruction-style prompts. We have an instruction-response pair for every training example. The tweet text acts as the input, and just one of the six labels is the expected output. This format is consistent with how these models are already taught to carry out instructions. It ensures that the labels output during testing are consistent.

Hyperparameter Settings

The PyTorch is model is used for the training purposes [5] and Hugging Face Transformers [4] as well. In order to maintain the integrity of comparisons, the same parameters are being used by each of the models:

• Max input length: 512 tokens • Batch size: 16 if needed (gradient accumulation) • Rate of learning: 1.5×10^{-5} plus warming and weight decay. • Pre-experiment: up to 15 with early stopping on validation loss (patience of 3 epochs). • LoRA setup: rank r=16, scaling α=32, dropout 0.05, focusing on attention projection modules, e.g., qkv and output projections.

The best validation loss checkpoint is used for the model selection.

Producing Predictions for Evaluation

With fine-tuning done, the test set predictions are generated using a restricted text output process. The model has to choose exactly one label from the defined categories. Here greedy decoding is employed. Plus the evaluation remains deterministic. All created text is normalized (case and extra whitespace). This is done before matching against the allowed label strings using a strict lookup.

Few-Shot In-Context Inference using Chain-of-Thought

Building Balanced Demonstrations

For few-shot inference, no parameter updates happens. A balanced sample of examples is assembled instead. It is done by drawing 12 samples per class (72 in total) from the training set. The sampling employs a fixed random seed to keep results reproducible. These demonstrations are made to enhance lexical variety. And, also to avoid repetitive phrasing within the prompt context.

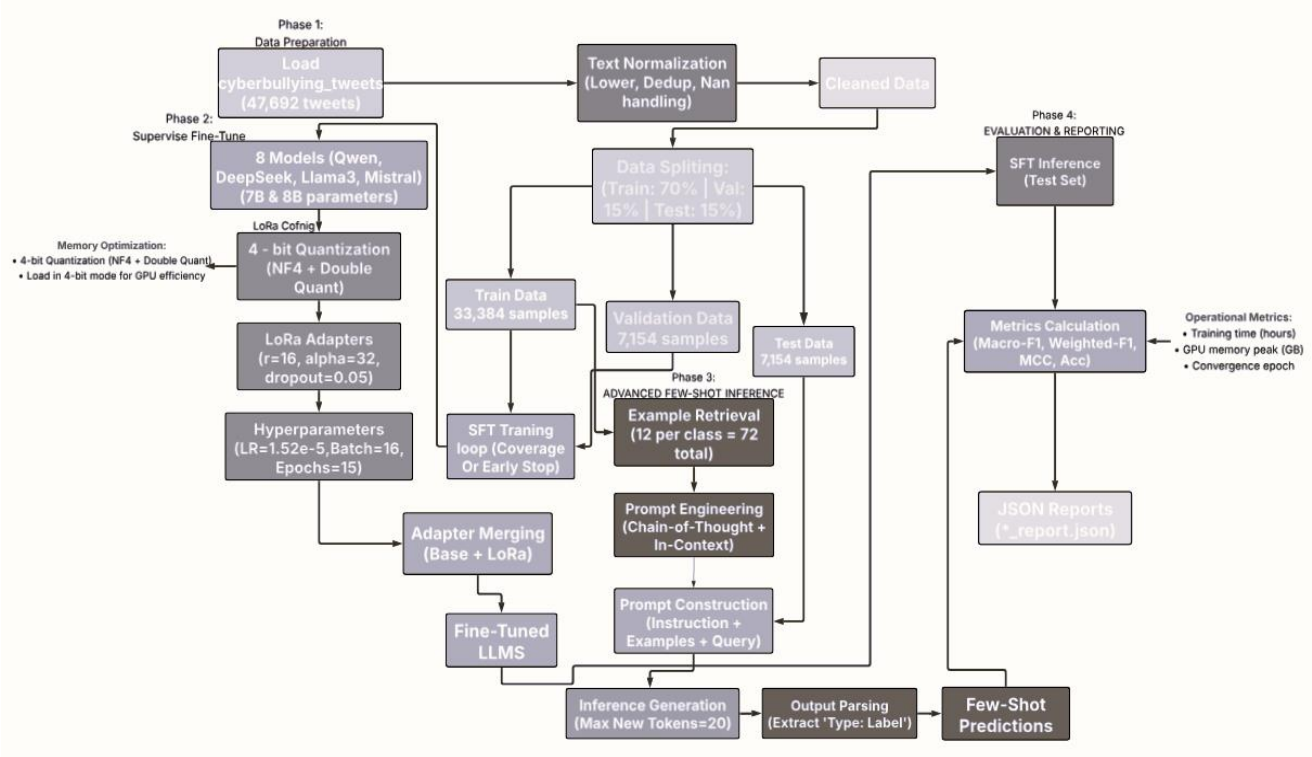

Figure 1: High-level overview of the cyberbullying detection pipelines, that shows the supervised fine-tuning with quantized LoRA adapters. It also tells about the few-shot in-context inference pathway.

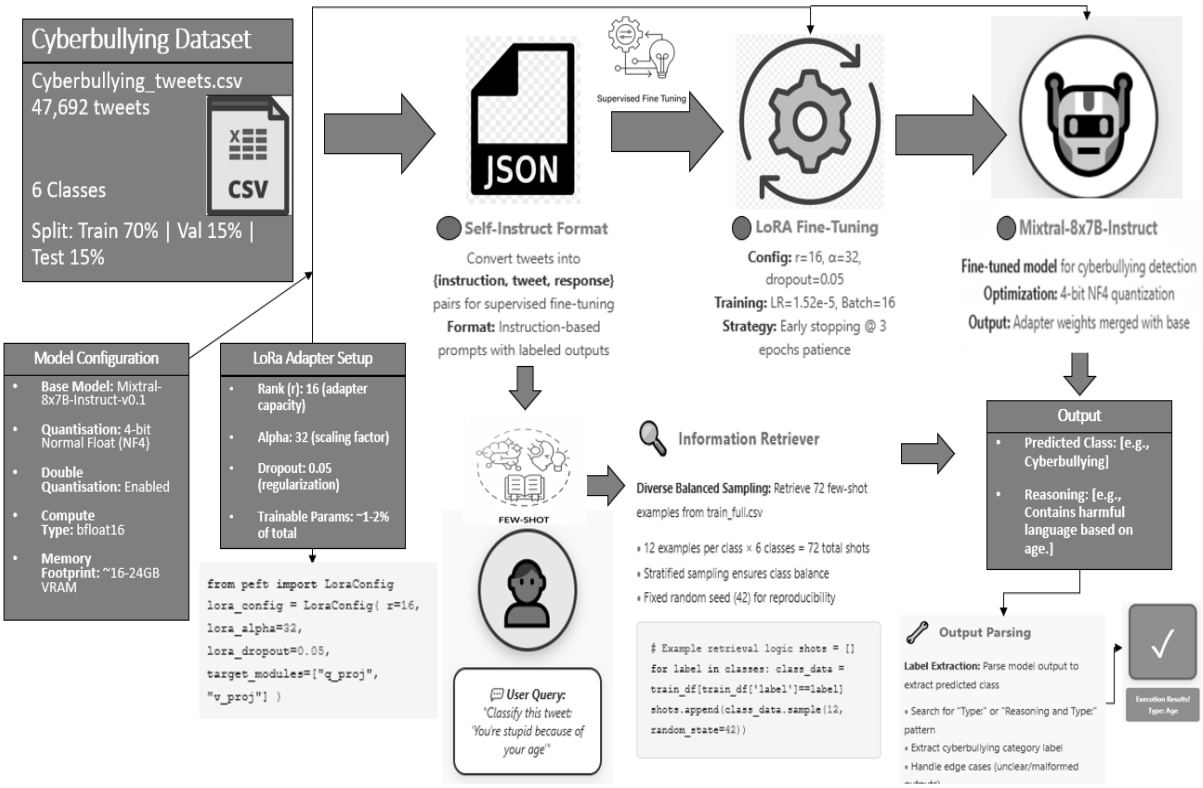

Figure 2: End-to-end experimental flow: data preparation, normalization, stratified splitting, parameter-efficient fine-tuning, inference, and reporting of predictive and operational metrics.

Prompt Structure

It begins with the some instructions, with examples like the same formating in Tweet and Reasoning and Type. After that, the actual tested tweet is tacked onto the bottom using that same layout. Chain-of-thought prompting is used to help the model better distinguish abuse categories that look a lot alike [15]. The final output is locked to just one of the six class labels.

Decoding and Parsing Outputs

Inference uses deterministic greedy decoding generation which stops any random variation from creeping in. The generated text is then parsed to find a single class label. A rule-based system looks for the very first instance of a valid label token. So, if the output doesn't contain a valid label string. The evaluation protocol marks that prediction as wrong (Section 5).

Implementation and Reproducibility Details

Every experiment plus the code was done in Python with PyTorch [5]. and Transformers [4]. The Scientific libraries handle the heavy lifting for data prep and analysis [6-9]. All training and testing runs on an NVIDIA RTX 3090 GPU (graphic card). Beyond just looking at how well the models predict, operational metrics-like how long training takes, what's the inference speeds is and peak GPU memory usage are all recorded. It is done to ensure others can replicate the work and compare deployment costs.

Evaluation Protocol

Evaluation Setup

In this paper, all results come from the held-out test partition mentioned in Section 3, using the same preprocessing and label categories. For the supervised fine-tuning part, the checkpoint

showing the lowest validation loss is the one picked for a single final test run. On the few-shot side, every backbone uses the same balanced demonstration set (12 samples per category) and deterministic decoding. This keeps the comparison between the two methods as tight as possible.

Prediction Extraction and Validity Rules

Both paradigms ultimately produce a single predicted label from the fixed label set: {not_cyberbullying, other_cyberbullying, age, ethnicity, gender, religion}. Predictions are obtained by decoding the model output and applying a strict parser that searches for the first occurrence of any valid label string. Output normalization includes lowercasing and whitespace trimming. If no valid label is found, the prediction is marked as incorrect for metric computation. This rule prevents the inflating scores via partial matches. It also avoids free-form generations and aligns evaluation across backbones and paradigms.

Classification Metrics

To get a full image of how these models works when it comes to class balance and general correctness. The Performance is tracked through several different lenses.

Accuracy

Accuracy is simply the percentage of test cases. The cases where the model got the label exactly right.

Precision, Recall, and F1-Score

The precision, recall, as well as the F1-score is also calculated for every individual class. When it comes to the overall total scores. The report of F1 is reported as:

• Macro-F1: Unweighted avg of per-class F1 scores (class-balanced). • Weighted-F1: avg of per/class F1 scores weighted by class support. • Micro-F1: calculated by globally synthesizing true positives, false positives, and false negatives across classes.

Matthews Correlation Coefficient (MCC)

This paper features the Matthews Correlation Coefficient (MCC), which is a widely accepted summary statistic for multi-class tasks that assesses the relationship between predicted and actual labels [13]. The classification is done in the classical multi-class generalization found in established evaluation toolkits.

Confidence Scores for Generative LLM Classifiers

Some metrics, like AUPRC and calibration error, need a confidence score for every prediction. Since both of our setups generate text strings rather than using a traditional softmax classification head, we have to derive these confidence levels through label likelihood scoring.

For every test input, we build a fixed prompt (formatted for either fine-tuning or few-shot). Let y be the set of six valid labels. For each label $y \in Y$, we calculate the conditional log-likelihood of generating that label's tokens given the model:

$$s(y) = \log p(y|x) \quad (1)$$

where x represents the formatted prompt (which includes demonstrations in the few-shot case). If a label consists of multiple tokens, the score is simply the sum of their log-probabilities.

A pseudo-probability distribution across the labels is subsequently derived by normalizing the scores via a softmax function:

$$\hat{p}(y|x) = \exp(s(y)) / \sum \exp(s(y')) \quad (2)$$

The predicted class is arg max $\hat{p}(y|x)$ while the corresponding confidence value is established as max $\hat{p}(y|x)$. By doing so the confidence estimates are consistent across all the backbones and do not refer by any value to free-form generation length and formatting irregularities.

Area Under the Precision-Recall Curve (AUPRC)

AUPRC is added to monitor precision recall information that a single decision threshold cannot measure [14]. In multi-class scenarios, one-vs-rest precision-recall curves are calculated for all of the classes c, treating c as the positive case and the other classes as negative with $\hat{p}(c|x)$ as the ranking metric. These per-class AUPRC numbers are then subjected to macro-averaging over the whole set:

$$AUPRC\_macro = (1 / |Y|) \sum AUPRC(c). \quad (3)$$

This macro-averaged formula represents a balanced view of performance across all six discrete classes.

Calibration plus the Expected Calibration Error (ECE)

Calibration calculates consistency of the predicted confidence levels with the help of real empirical accuracy. Expected Calibration Error (ECE) is computed using a standard estimation approach, that relies on binning [12]. This involves splitting predictions into M confidence bins with each having equal width in the [0, 1] range. For each bin B_m, accuracy and mean confidence are obtained:

$$aec(B_n) = (1 / |B_n|) \sum 1(\hat{y}_i = y_i), \quad (4)$$

$$conf(B_n) = (1 / |B_n|) \sum \max \hat{p}(y|xx_i). \quad (5)$$

The ECE is after derived:

$$ECE = \sum (|B_n| / N) |aec(B_n) - conf(B_n)|. \quad (6)$$

A smaller ECE tells a better match between the confidence scores and the observed accuracy. This alignment is important for moderation workflows based on thresholding, or for human review.

Operational Metrics

Apart from raw predictive accuracy, operational stats are included to convey a glance of deployment costs:

• Training time (hours) needed to use supervised fine-tuning, as well as the inference time (seconds) necessary for few-shot inference. • Throughput (tokens/s) in few-shot inference context. • Peak GPU memory (GB) used in training and inference.

Such figures give a view of the tradeoff of compute resources verses performance. They sit along the classification quality to guide practical decisions on which system to choose.

Results and Discussion

Overall Results: Supervised Fine-Tuning

Table 2 reports the results for supervised fine-tuning (SFT) including eight instruction-tuned LLM backbones that adopted the same QLoRA-style parameter-efficient mechanism (Section 4). Since the dataset has a fairly balanced split across classes (Section 3), accuracy and macro-F1 generally track closely together. This provides a good indication of how the model balances the amount of classes.

Across the backbones, the macro-F1 scores are between 0.809 and 0.857. It turns out that the SFT setup's best performer is Mixtral-8x7B-Instruct, which delivers macro-F1 of 0.857 and accuracy of 0.857. It also posts strong reliability numbers (MCC 0.808, AUPRC 0.911) with remarkably low calibration error (ECE 0.021). For the dense 7B/8B group, Qwen2.5-8B-Instruct holds its own (macro-F1 0.845), as does Mistral-7B-Instruct (macro-F1 0.841) and DeepSeek-8B-Instruct (macro-F1 0.833), which come in just behind. These results imply that adapting models with an

appropriate approach that is efficient with respect to parameters consistently provides high-quality classifiers for fine-grained cyberbullying detection, regardless of the specific model family.

Table 2: Supervised fine-tuning (SFT) results on the 6-class test set. These metrics reflect single-GPU runs at RTX 3090.

Models	Macro-F1	Accuracy	MCC	AUPRC	ECE	Train (h)	Peak GB	Conv. Ep.	Best Val Loss
Qwen2.5-7B-Instruct	0.830	0.830	0.785	0.894	0.029	25.0	18.2	7	0.392
Qwen2.5-8B-Instruct	0.845	0.845	0.801	0.906	0.023	28.7	19.8	6	0.368
DeepSeek-7B-Instruct	0.816	0.816	0.763	0.880	0.035	22.3	17.6	8	0.423
DeepSeek-8B-Instruct	0.833	0.833	0.781	0.892	0.027	26.1	18.9	7	0.381
Llama-3-7B-Instruct	0.809	0.809	0.754	0.876	0.038	21.6	17.2	9	0.433
Llama-3-8B-Instruct	0.824	0.824	0.770	0.885	0.032	24.2	18.1	8	0.404
Mistral-7B-Instruct	0.841	0.841	0.794	0.901	0.024	26.5	18.7	6	0.376
Mixtral-8x7B-Instruct	0.857	0.857	0.808	0.911	0.021	30.0	20.1	5	0.352

Overall Results: Few-Shot In-Context Inference

Table 3 provides a summary of the results for few-shot in-context inference. This system employed 12 examples per category (72 total), added to this process in combination with chain-of-thought prompting (Section 4). Few-shot prompting generally lags behind SFT but still puts up a fight. Macro-F1 scores range between 0.770 and 0.820, the best performance again claimed by Mixtral-8x7B-Instruct (macro-F1 0.820). Overall, across all matched backbones, the gap between SFT and few-shot averages roughly 0.037 (between 0.035-0.039). And that means few-shot prompting catches a tremendous amount of fine-tuned performance without requiring a single gradient adjustment.

When one considers operations, the full test set was processed in 1,732-2,286 s, depending on the backbone. At length, throughput ranged from 26.7 to 33.2 tokens/s. The average prompt lengths betray a trade-off. Actually, those 72 demos consume a big chunk of the context window. That has real ramifications for both latency and cost.

Table 3: Few-shot + chain-of-thought (CoT) results on the 6-class test set with 72 in-context demonstrations (12 per class).

Models	Macro-F1	Accuracy	MCC	AUPRC	ECE	Train (h)	Peak GB	Conv. Ep.	Best Val Loss
Qwen2.5-7B-Instruct	0.792	0.792	0.729	0.846	0.045	1872.5	30.1	17.5	1568
Qwen2.5-8B-Instruct	0.810	0.810	0.751	0.863	0.040	2135.8	27.8	19.2	1630
DeepSeek-7B-Instruct	0.778	0.778	0.712	0.832	0.052	1783.8	32.5	16.5	1523
DeepSeek-8B-Instruct	0.798	0.798	0.737	0.851	0.046	2035.1	29.4	18.1	1576
Llama-3-7B-Instruct	0.770	0.770	0.703	0.825	0.056	1732.3	33.2	16.0	1478
Llama-3-8B-Instruct	0.786	0.786	0.723	0.840	0.049	1947.6	30.8	17.5	1538
Mistral-7B-Instruct	0.804	0.804	0.745	0.857	0.041	2081.4	29.0	18.4	1596
Mixtral-8x7B-Instruct	0.820	0.820	0.764	0.871	0.036	2285.9	26.7	19.8	1655

Per-Class Performance and Confusion Trends

To emphasize class-specific behavior, Table 4 reports per-class precision/recall/F1. For a better-fitting fine-tuned model (Qwen2.5-8B-Instruct) and Table 5 gives a per-class F1. For the best few-shot model (Mixtral-8x7B-Instruct). Although all the data comes from both worlds. The not_cyberbullying has at least one more F1, indicating that the lexis is cleaner, and less murky in terms of ambiguous statements. By contrast, other_cyberbullying is always more difficult, probably because it is more uniform, compressing multiple types of bullying styles over different categories. Which overlaps a little across various identity-targeted abuse types. This overlap may

give rise to systematic confusions at the decision boundary, prompting reporting per class metrics along with overall scores.

Table 4: Every classes per results for Qwen2.5-8B-Instruct under supervised fine-tuning (SFT).

Class	Prec.	Rec.	F1
not_cyberbullying	0.922	0.906	0.914
other_cyberbullying	0.799	0.822	0.810
age	0.855	0.848	0.851
ethnicity	0.831	0.815	0.823
gender	0.864	0.853	0.858
religion	0.844	0.831	0.837

Table 5: Per-class F1 for Mixtral-8x7B-Instruct under few-shot + CoT inference.

Class	F1
not_cyberbullying	0.874
other_cyberbullying	0.771
age	0.804
ethnicity	0.782
gender	0.823
religion	0.797

Reliability and Calibration

Accuracy doesn't tell the full story. The calibration provides an in-depth look. As SFT shows, ECE values vary between 0.021 and 0.038. With Mixtral-8x7B-Instruct it is showing the tightest calibration (0.021). In contrast, few-shot inference yields higher ECE values across the board

(0.036 to 0.056). This appears to show that the confidence estimates are inferior even when it is reasonable to forecast accurately. In real-world moderation workflows, where thresholds of confidence influence. When to trigger a human review, this difference is important.

Compute-Performance Trade-Off

SFT makes the strongest predictions, but is the most resource-heavy method. It requires a lot of training (21.6-30.0 hours per model), and drives peak GPU memory to 17.2-20.1 GB. Few-shot inference bypasses the training grind altogether, running directly on base backbones. The catch? It's about long prompts and obvious inference latency due to that 72-shot context. The average macro-F1 gap of ≈ 0.037 lays down this trade-off in numbers: fine-tuning gets gains, while few-shot prompting continues to be a solid go for quick rollouts, or places where training is out of reach.

Finally, parameter-efficient supervised fine-tuning reaches the optimal trade-off between accuracy and reliability for fine-grained cyberbullying classification. Few-shot prompting seems like a good fit: it approaches fine-tuned performance, all with no training overhead. However, it has weaker calibration and generally costs more per prompt.

Conclusion and Future Work

This paper established a clear comparison of transformer-based learning approaches. Also, for the detection of fine-grained cyberbullying. Six-class social media dataset was employed in the experiments and a well-specified and repeatable model implemented in a highly monitored environment. Two pragmatic paradigms were also evaluated, with similar stratified splits and the same preprocessing: (i) parameter-efficient supervised fine-tuning (SFT) by means of 4-bit quantization (as well as LoRA adapters). (ii) few-shot in-context inference with balanced demonstrations and chain-of-thought prompting. When tested across eight instruction-tuned LLM backbones, supervised fine-tuning produced macro-F1 scores in the range of 0.809-0.857; accuracy hovered somewhere around this same 0.809-0.857 threshold. Few-shot inference wasn't far behind - macro-F1s ranged between 0.770 and 0.820. And yet it had a catch: lower calibration and higher inference costs based on the size of our prompts, even without the training phase.

Several future research directions are still open to explore and not have been presented in the present paper. First, we will then consider robustness under distribution shift which entails, respectively, temporal drift and cross-platform transfer in order to more closely simulate the disorganized behavior of actual moderation. Second, the speed at high resolution and sensitivity for few-shot inference require a systematic review that uses varying combinations of patterns in the selection, order and constraints of demonstrations. Third, comparisons are increased; additional baseline families such as encoder-only transformers (BERT, ROBERTa, DeBERTa) and classical feature-based models will be added to enhance the performance of the cross-paradigm correlation analysis. Lastly, fairness and bias assessments are anticipated. This step will test whether the error rates vary considerably between identity-based types, which is crucial for the safe implementation of fine-grained cyberbullying detection systems.

References

[1] X. Wang, K. Fu, and C. Lu, "SOSNet: A graph convolutional network approach to fine-grained cyberbullying detection," in Proc. IEEE Int. Conf. Big Data (Big Data), 2020.

[2] P. Yi, A. Zubiaga, and Y. Long, "Detecting harassment and defamation in cyberbullying with emotion-adaptive training," in Proc. Int. AAAI Conf. on Web and Social Media (ICWSM), 2025. (Also available as a PDF from the authors.)

[3] AndrewMvd, "Cyberbullying tweets dataset (6-class)," Kaggle, [Online]. (Accessed 2025).

[4] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Transformers: State-of-the-art natural language processing," in Proc. EMNLP 2020: System Demonstrations, 2020.
[5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems (NeurIPS), 2019.
[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[7] W. McKinney, "Data structures for statistical computing in Python," in Proc. Python in Science Conf. (SciPy), 2010.
[8] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," Nature, vol. 585, pp. 357-362, 2020.
[9] J. D. Hunter, "Matplotlib: A 2D graphics environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
[10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LORA: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.
[11] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLORA: Efficient finetuning of quantized LLMs," arXiv preprint arXiv:2305.14314, 2023.
[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in Proc. Int. Conf. on Machine Learning (ICML), 2017.
[13] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," Biochimica et Biophysica Acta, vol. 405, no. 2, pp. 442-451, 1975.
[14] J. Davis and M. Goadrich, "The relationship between Precision-Recall and ROC curves," in Proc. Int. Conf. on Machine Learning (ICML), 2006.
[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," arXiv preprint arXiv:2201.11903, 2022.
[16] T. B. Brown, B. Mann, N. Ryder, J. Kaplan, P. Dhariwal, and others, "Language models are few-shot learners," in Advances in Neural Information Processing Systems (NeurIPS), 2020.
[17] L. Ouyang, J. Wu, X. Jiang, D. Almeida, and others, "Training language models to follow instructions with human feedback," arXiv preprint arXiv:2203.02155, 2022.
[18] J. Wei et al., "Finetuned language models are zero-shot learners (FLAN)," arXiv preprint arXiv:2109.01652, 2021.
[19] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019.
[20] Y. Liu, M. Ott, N. Goyal, and others, "ROBERTa: A robustly optimized BERT pretraining approach," arXiv preprint arXiv:1907.11692, 2019.
[21] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with disentangled attention," in Proc. Int. Conf. on Learning Representations (ICLR), 2021.
[22] A. Conneau, K. Khandelwal, N. Goyal, and others, "Unsupervised cross-lingual representation learning at scale," in Proc. ACL, 2020.
[23] M. Zampieri, S. Malmasi, P. Nakov, and others, "Predicting the type and target of offensive posts in social media," in Proc. NAACL-HLT (OffensEval/OLID), 2019.

[24] T. Mandl, S. Modha, P. Majumder, and others, "Overview of the HASOC track at FIRE 2019," in Proc. FIRE, 2019.
[25] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in Proc. ICWSM, 2017.
[26] Z. Waseem and D. Hovy, "Hateful symbols or hateful people?" in Proc. NAACL Workshop on Abusive Language, 2016.
[27] A.-M. Founta, C. Djouvas, D. Chatzakou, and others, "Large scale crowdsourcing and characterization of Twitter abusive behavior," in Proc. ICWSM, 2018.
[28] B. Vidgen and L. Derczynski, "Directions in abusive language training data: Garbage in, garbage out," arXiv preprint arXiv:2004.01670, 2020.
[29] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," ACM Computing Surveys, vol. 51, no. 4, pp. 1-30, 2018.
[30] Jigsaw and Conversation AI, "Toxic comment classification resources and benchmarks," 2018-2019.
[31] Meta AI, "Meta Llama 3 model card and release documentation," 2024.
[32] Qwen Team, "Qwen2.5 Instruct model cards and technical documentation," 2024.
[33] DeepSeek AI, "DeepSeek large language model technical report and documentation," 2024.
[34] Mistral AI, "Mistral and Mixtral model cards and release documentation," 2023-2024.