NLP CS6501-011 Project Report
Hate Speech Detection and Classification Using Multi LLM Architecture

Sravanth Chowdary Potluri
und5uv@virginia.edu

Darsh Naresh Jain
bhj4jy@virginia.edu

Asjad Sajid Nirban
wjw8yd@virginia.edu

1 Introduction

1.1 Problem Description

In recent years, the rise of online social media platforms has led to an increase in harmful and offensive content, particularly hate speech. Hate speech can take various forms, including racism, sexism, and other discriminatory language, leading to toxic environments that harm public discourse. With millions of interactions through natural language taking place on online social media daily, The ability to automatically detect and categorize hate speech is crucial for promoting healthy digital environments and preventing the spread of harmful ideologies for the overall social well being and reducing toxicity on the internet in general.

1.2 Motivation

Social Media has a significant influence on our lives. The fact that a lot of people we know, including us, open Instagram, X, Reddit etc right after waking up from sleep is testament to the influence of Social Media. One of the key components of Social media is interaction via text(Natural Language) and these might contain hate speech as discussed in the previous section. The fact that hate speech is so prevalent in the online domain which influences millions of people regularly is a genuine concern for us and we would like to try to put our Knowledge of NLP to use by providing a solution for this issue. (Alkomah and Ma, 2022)

1.3 NLP Formulation of the Problem

This problem can be formulated as an NLP task in two ways. The first way is to perform classification/semantic analysis to identify whether a given string can be classified as a hateful comment or not and if the string is found to be hateful, what subclass of hate speech does the given text fall under such as Animosity, Derogation, Dehumanization, Threatening, Support for Hateful Entities etc. This Classification and Categorization can offer Social Medial organizations a way to Restrict/Remove Hateful interactions online or Redact the hateful components of a comment or a tweet to ensure safer online interactions. There might also be different policies to deal with different kind of Hate Speech.

2 Data

2.1 Dataset Origin

The Dataset we are going to be using for this task is a publicly available dataset generated as a part of (Vidgen et al., 2021). This dataset was generated through a human-and-model-in-the-loop process over four rounds. In each round, annotators created adversarial content to trick a model, and a new model was trained using this data. The process involved annotators generating original content and challenging "contrast sets" (perturbations of existing data), progressively increasing the difficulty of the classification dataset, so that the models can learn better. The dataset we will be using is the final and refined dataset obtained from the process.

2.2 Dataset Description

The version of the dataset we will be using contains 41144 samples in total with key fields such as 'Text' Which contains the content to be classified. There is a 'Label' Field which contains the data whether the given text is Hateful or not and a 'Type' field which contains lower level information about the type of hate pertaining to the given sample. Overall the samples contain around 22175(54%) hateful samples, with the lower level classes containing a significant number of samples in each of the classes such as Derogation,Animosity and Dehumanisation with 9907(24%),3439(8%) and 906(2%) Samples respectively. The Training, testing and validation samples as classified by the authors are 32924(80%), 4120(10%) and 4100(10%) samples respectively. Coming to the statistics of the textual part of the data, The text in the dataset contains 1001530 tokens in total with a vocabulary size of 138677 words.

3 Proposed Approach

3.1 Solution Description

In the field of Natural Language Processing, Hate Speech detection and classification is a quite popular task. (Alkomah and Ma, 2022) states that there are an average of 30 works on hate speech detection per year. And around 70% of these make use of neural networks in various forms such as (RNN's,CNN's etc) for this task. However, Large Language Models(LLM's) which are on the rise in the field of NLP right now are still rare for this task. In the project, we propose a use of

Large Language Models for this task of classification which aren't that widely used in This task. However, Instead of using just a single LLM. We would like to make use of two LLM's in varying sizes for this task of classification of Hate speech.

- **LLM 1: Hate Speech Identification (Smaller LLM)** The first LLM we use is going to be a smaller LLM of the two example Meta's Llama 3.2's 1 Billion parameter model. We would like to use this LLM for a simpler task of binary classification i.e whether the given text contains hate speech or not. By using this smaller model for initial screening, we aim to quickly discard non-hateful content without invoking the more resource-intensive model.
- **LLM 2: Hate Speech Categorization** We make use of the second and Larger LLM such as Meta's Llama 3.2's 3 Billion parameter model. We are going to use this model for a more complex task of categorization of which subcategory of hate speech the given text falls under. The usage of this model is only triggered when the given text is classified as Hateful by the previous smaller model.This step leverages the larger LLM's better language understanding and contextual analysis, ensuring precise categorization of the hate speech when necessary.

Through this project, we would like to analyse and investigate the benefit of using this Multi LLM architecture and the different nuances involved in this approach. We plan to fine-tune these LLM's through the Training set of our dataset and the evaluate the performance of these models .Our working hypothesis is that our approach avoids invoking the large LLM for every piece of text, thus saving computational time and resources, particularly when processing large datasets where the majority of content is non-hateful.

3.2 Evaluation Criteria & Metrics

To assess the effectiveness of our proposed approach, we will evaluate it using the following criteria:

- **Accuracy**: We will measure the *accuracy* of both the hate speech identification and categorization tasks. For identification, we will compare how accurately the small LLM can detect hate speech against a ground truth dataset. For categorization, we will evaluate how well the large LLM categorizes hate speech into the correct predefined categories.
- **Time Efficiency**: Another metric of interest is the *average runtime* per text for both methods. Our two-stage model versus the traditional single large LLM approach. This will include the total processing time for both identification and categorization tasks. We will also find the average of runtime of using only the larger LLM for classification and compare these two results under different scenarios of varying percentage of hateful samples to provide a comprehensive overview on the Trade-offs of this approach.
- **Computational Cost**: We will also track the *resource utilization*, such as CPU/GPU usage and memory consumption, during the execution of both the dual-LLM approach and the single-LLM baseline. This will help quantify the computational load when using the different structure of LLM's

Training and deployment of LLM's for any task could be a cost and computationally intensive task. Even if you are just using the API, it could turn out to be expensive and this proposed approach with these metrics investigates whether this approach can turn out to be efficient in the real world scenario where the number of interactions with hate speech are generally low compared to the overall number of interactions on Social Media.

4 Methodology And Implementation

4.1 Contribution

The project tackles the crucial task of detecting and categorizing hate speech in online social media interactions. Given the significant computational resources required for large-scale text classification, the approach adopts a novel two-tiered multi-LLM (Large Language Model) architecture. This system integrates two models of varying sizes—Meta's LLaMA 3.2 with 1 billion parameters and Meta's LLaMA 3.2 with 3 billion parameters—each performing distinct roles to achieve efficiency and precision.

The smaller LLM (1B parameters) focuses on binary classification, identifying whether a given text contains hate speech. If the text is flagged as hateful, the larger LLM (3B parameters) is invoked to categorize it into specific subcategories, such as dehumanization, derogation, or animosity etc. This tiered approach optimizes computational resources by reserving the larger and more resource-intensive model for the nuanced task of hate speech categorization, reducing unnecessary overhead.

To assess the performance of this two-stage architecture, we also fine-tuned the larger 3B model to perform the entire task of classification and categorization in one step. This setup serves as a baseline for comparison, allowing us to analyze the trade-offs between computational efficiency and performance.

The major contribution of this project lies in the development of this multi-LLM architecture, which demonstrates that a two-stage filtering and categorization system can maintain accuracy while significantly reducing computational costs. By leveraging fine-tuning and in-context learning, the approach aligns large-scale model performance with real-world constraints, offering a scalable solution for hate speech detection in social media contexts.

4.2 Finetuning And Inference

Fine-tuning plays a central role in adapting both LLMs to the specific tasks of hate speech detection and categorization. The process involves the use of custom-designed prompts containing examples that guide the models to understand the tasks effectively.

4.2.1 Smaller Model: Binary Classification

The smaller model classifies the input text into one of two outputs: "hate" for hate speech and "nothate" for non-hate speech. The task is simplified to binary classification, and the prompt is structured as follows:

Given the text, predict whether the text is hate speech or not. You can only output one word, either 'hate' or 'nothate.' For example:<Text 1> -> hate, <Text 2> -> hate, <Text 3> -> nothate, <Text 4> -> nothate, <Text 5> -> hate, <Text 6> -> nothate, Now the Text:

4.2.2 Larger Model: Fine-Grained Categorization of Hate Speech

Once the smaller model identifies a text as hate speech, the larger model categorizes the hate speech into specific subcategories: derogation, animosity, dehumanization, threatening, or support. The fine-tuning prompt for the larger model is structured as follows:

Given the text, which is hate speech, predict which kind of hate speech the text falls under. Your output can only be one word. You can output one of the following categories: derogation, animosity, dehumanization, threatening, support. For example: <Text 1> -> derogation, <Text 2> -> animosity, <Text 3> -> dehumanization, <Text 4> -> threatening, <Text 5> -> support, Now the Text:

4.2.3 Baseline Model: Combined Classification and Categorization

To compare the efficiency of the two-tier architecture, the larger model is fine-tuned to handle both binary classification and categorization simultaneously. The prompt combines elements from the smaller and larger model prompts.

Given the text, predict whether the text is hate speech or not and, if it is hate speech, which category of hate speech it falls under. Your output can only be one word. If the text is not hate speech, output 'nothate.' If it is hate speech, output one of the following categories: derogation, animosity, dehumanization, threatening, support. For example: <Text 1> -> nothate, <Text 2> -> derogation, <Text 3> -> animosity, <Text 4> -> nothate, <Text 5> -> dehumanization, <Text 6> -> threatening, Now the Text:

The project faced several challenges during fine-tuning, particularly with the 3B model. Training this larger model required approximately one hour per epoch, and training for more than one epoch caused the loss to diverge to NaN (not-a-number). This issue likely stemmed from overly aggressive updates to model weights or large gradients during optimization, which destabilized the training process. To mitigate this, the 3B model was trained for only one epoch. In contrast, the smaller 1B model, which required only 20 minutes per epoch, was trained for five epochs. This disparity in training duration was another factor driving the emphasis on leveraging ICL to compensate for the reduced training time for the larger model.

4.2.4 In-Context Learning For Finetuning And Inference

In-context learning (ICL) is a methodology that allows a language model to perform tasks by embedding labeled examples directly within the input prompt, guiding the model to understand and execute the task without requiring extensive retraining. Unlike traditional supervised learning, where models are trained on a large dataset, ICL leverages the model's pre-trained capabilities and provides a set of examples as context. The model then generalizes from these examples and aligns its output to the patterns demonstrated within the prompt.

In this project, in-context learning was used during both fine-tuning and inference. While traditionally ICL is employed only during inference, its inclusion in the fine-tuning phase proved crucial for enhancing the models' task-specific understanding. During fine-tuning, prompts were designed with labeled examples that clearly illustrated the expected behavior for binary classification and categorization tasks. For instance, in the case of the smaller model, the prompt demonstrated how the model should differentiate between hate speech and non-hate speech by providing example pairs of input sentences and their corresponding outputs. Similarly, the larger model was fine-tuned with prompts containing examples of hate speech and their specific categories, enabling it to specialize in fine-grained categorization.

4.3 Hyper-parameter Tuning

Hyper-parameter tuning was essential for optimizing the performance of the models during inference in this project. Given the time-intensive nature of fine-tuning, it was impractical to experiment with different hyper-parameter combinations during training. Instead, the tuning process focused on the inference phase, where adjustments to critical parameters such as Top-K sampling and temperature were more feasible and efficient.

Top-K Sampling

Top-K sampling was one of the primary hyper-parameters explored during inference. This parameter plays a crucial role in controlling the range of tokens considered by the model during prediction by truncating the output probability distribution to the top K most likely tokens. This ensures that the model focuses only on the most relevant and probable outputs.

In this project, three values for Top-K were tested: 300, 10, and 1. A higher Top-K value (e.g., 300) allowed

the model to consider a broader range of possible outputs, which could be useful for creative or exploratory tasks. However, this approach introduced unnecessary diversity and noise, leading to inconsistent and less reliable predictions for deterministic tasks like binary classification and categorization. Conversely, lower values (e.g., 10 or 1) restricted the model to a narrower range of options. While this could potentially limit diversity, it resulted in more focused and reliable outputs.

Temperature

Temperature is another critical hyper-parameter that influences the randomness of the model's predictions. This parameter scales the logits (raw outputs) before applying the softmax function to compute probabilities, effectively controlling how likely the model is to explore less probable options.

In this project, three temperature values were tested: 0.6, 0.3, and 0.001. A higher temperature (e.g., 0.6) increased randomness in the predictions, allowing the model to explore a wider range of outputs. While this approach could be beneficial in creative tasks, it introduced variability and decreased reliability in deterministic tasks like hate speech detection. Lower temperatures (e.g., 0.3 or 0.001) made the predictions more deterministic by amplifying the difference between high and low probability tokens.

4.3.1 Packages Used

The implementation of this project heavily relied on the Torchtune library for both fine-tuning and inference. Torchtune provided an efficient framework to adapt pretrained large language models (LLMs) using a concept called "recipes", which takes care of everything in fine-tuning and inference given a proper configuration .yaml file. However we had to customize the inference recipe a bit to adapt it to inference from a dataset without loading the model everytime and give out the required statistics such as inference time and memory usage.

4.4 Source Code

All the code for the project including the Jupyter Notebooks, The Config Files for finetuning, inference and the input,generated datasets are available on Github. The finetuned model files are not uploaded because of their size, however they can be produced upon request by uploading on hugging face.

5 Results

All the results with different hyper parameters are compiled in 1 with the results of models that are part of the architecture along with the baseline 3B Model for comparison.

5.1 Result Analysis

5.1.1 Effective Time and Memory Usage

The proposed two-tier multi-LLM architecture for hate speech detection and categorization demonstrates significant improvements in computational efficiency under the assumption that 5% of content on social media is hateful. The key results from experiments highlight the following:

- Time Efficiency: The architecture achieved approximately 18% improvement in processing time compared to using a single large model (LLaMA 3B) for all samples. This is primarily because the smaller LLaMA 1B model is employed for initial binary classification, avoiding the resource-intensive larger model for the vast majority of non-hateful content.

- Memory Efficiency: The two-tiered approach reduced memory usage by approximately 54%. This is attributed to the fact that the larger LLaMA 3B model is invoked only for hate speech categorization, significantly lowering the memory footprint for non-hateful content.

To derive the effective time and memory consumption for the multi-LLM architecture, we used the following formulae:

$$T_{eff} = P_{hate}(T_{Small} + T_{Big}) + (1 - P_{hate})T_{Small}$$
$$M_{eff} = P_{hate}(M_{Small} + M_{Big}) + (1 - P_{hate})M_{Small}$$

Where:

- Where T_{eff} and M_{eff} are the effective times and memory of the multi llm architecture

- Where P_{hate} is the percentage of hate samples

- Where T_{Small},T_{Big},M_{Small},M_{Big} are the inference times and memory usages of the smaller and bigger models in the multi LLM architecture.

5.1.2 Observations and Insights from the Table

From the table, we can infer the following:

- Classification Accuracy (LLaMA 1B): Across all hyperparameters tested, the smaller LLaMA 1B model consistently achieved 78.1% accuracy for binary classification. This demonstrates that the smaller model is decently reliable for the initial identification of hate speech.

- Categorization Accuracy (LLaMA 3B): Given that hate speech was identified, the larger LLaMA 3B model achieved 69% accuracy under the most deterministic setting (Temp = 0.001, Top_K = 1). However, when considering all input content, the overall categorization accuracy drops to 64% due to non-hateful samples being included.

- 3. Resource Usage:
  • Time: The mean time for binary classification (LLaMA 1B) is approximately 0.058 seconds per sample, while categorization (LLaMA 3B) requires 0.118 seconds for hateful samples. The proposed two-tier system avoids invoking the larger model unnecessarily for the majority of inputs since the larger model clearly requires more computation

Table 1: Results Of Classification On Llama 1B and Llama 3B Models

Model	Metric/Hyperparameter	Temp = 0.6, Top_K = 300	Temp = 0.3, Top_K = 10	Temp = 0.001, Top_K = 1
LLaMA 1B(Multi LLM)	Classification Accuracy	0.781	0.784	0.784
	Mean Classification Time (s)	0.058	0.059	0.058
	Mean Classification Memory (GB)	2.92	2.92	2.92
LLaMA 3B Given Hate (Multi LLM)	Categorization Accuracy (Given Hate)	0.65	0.68	0.69
	Mean Categorization Time (Given Hate) (s)	0.112	0.114	0.118
	Mean Categorization Memory (Given Hate) (GB)	6.995	6.995	6.995
LLaMA 3B Baseline	Categorization Accuracy	0.60	0.64	0.64
	Mean Categorization Time (s)	0.077	0.076	0.076
	Mean Categorization Memory (GB)	7.15	7.15	7.15

time.

• Memory: The LLaMA 1B model consumes 2.92 GB for classification, whereas the LLaMA 3B model consumes approximately 7 GB for categorization. The proposed two-tier system reduces the memory burden significantly for non-hateful content since it uses the smaller model most of the time.

5.1.3 Context-Based Performance and Limitations

While the results in time and memory consumption are promising, they are heavily context-dependent. The performance improvements are significant under the assumption that hate speech comprises only 5% of the data. In scenarios where hateful content is more prevalent, the computational benefits would diminish as the larger LLaMA 3B model would be invoked more frequently.

Additionally, this analysis does not account for the combined accuracy of the multi-LLM architecture. The sequential nature of the architecture introduces dependencies:

- If the smaller LLaMA 1B model misclassifies hateful content as non-hateful, the larger model will not be invoked, leading to missed detections.
- The combined accuracy of the two-tier system may therefore be worse than a single, end-to-end large LLaMA 3B model.

Overall, The results are expected and good, with the success primarily driven by the two-tier architecture that leverages the smaller LLaMA 1B model for initial binary classification. This allows for efficient filtering of non-hateful content, significantly reducing the need to invoke the larger LLaMA 3B model for categorization. As a result, the system achieves an 18% improvement in processing time and a 54% reduction in memory usage compared to using the larger model for all samples. However, a notable issue lies in the potential reduction of combined accuracy for the overall system. If the smaller model misclassifies hateful content as non-hateful, the larger model is not triggered, leading to missed categorizations and a performance gap compared to an end-to-end single large model.

6 Conclusion and Future Scope

In this project, we proposed a two-tiered multi-LLM architecture for hate speech detection and categorization, leveraging a smaller model for binary classification and a larger model for fine-grained categorization. The results demonstrated significant improvements in time efficiency (18%) and memory usage (54%), particularly when hateful content constitutes a small fraction of social media data, such as 5%. These findings highlight the potential of hierarchical filtering approaches to reduce computational costs without major sacrifices in performance. However, the approach's success is context-dependent, and the combined accuracy of the multi-LLM system could be a limitation compared to a single, end-to-end large model. Future work can address this by improving the smaller model's accuracy through calibration techniques and error mitigation strategies to reduce false negatives. Efficient fine-tuning methods like LoRA and QLoRA can optimize resource usage for larger models, while approaches such as zero-shot and few-shot learning offer promising alternatives to full fine-tuning. Exploring ensemble approaches with multiple small LLMs can further improve performance, interpretability, and efficiency. To adapt the system to real-world scenarios, adaptive thresholds can be implemented to handle varying distributions of hateful content, while metrics for energy efficiency can assess environmental impact. This work establishes a strong foundation for scalable, resource-efficient NLP systems and offers promising opportunities for extending the architecture to other hierarchical classification tasks, such as multi-stage content filtering or topic categorization, across various domains.

References

Fatimah Alkomah and Xiaogang Ma. 2022. A literature review of textual hate speech detection methods and datasets. *Information*, 13(6):273.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection. 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 1667–1682, Online. Association for Computational Linguistics.