Bridging the Safety Gap: A Guardrail Pipeline for Trustworthy LLM Inferences

Shanshan Han
University of California, Irvine
Irvine, California, USA
shanshan.han@uci.edu

Salman Avestimehr
University of Southern California
Los Angeles, California, USA
avestime@usc.edu

Chaoyang He
TensorOpera AI
Palo Alto, California, USA
ch@tensoropera.com

Abstract

We present Wildflare GuardRail, a guardrail pipeline designed to enhance the safety and reliability of Large Language Model (LLM) inferences by systematically addressing risks across the entire processing workflow. Wildflare GuardRail integrates several core functional modules, including Safety Detector that identifies unsafe inputs and detects hallucinations in model outputs while generating root-cause explanations, Grounding that contextualizes user queries with information retrieved from vector databases, Customizer that adjusts outputs in real time using lightweight, rule-based wrappers, and Repairer that corrects erroneous LLM outputs using hallucination explanations provided by Safety Detector. Results show that our unsafe content detection model in Safety Detector achieves comparable performance with OpenAI API, though trained on a small dataset constructed with several public datasets. Meanwhile, the lightweight wrappers can address malicious URLs in model outputs in 1.06s per query with 100% accuracy without costly model calls. Moreover, the hallucination fixing model demonstrates effectiveness in reducing hallucinations with an accuracy of 80.7%.

ACM Reference Format:
Shanshan Han, Salman Avestimehr, and Chaoyang He. 2025. Bridging the Safety Gap: A Guardrail Pipeline for Trustworthy LLM Inferences. In . ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Large Language Models (LLMs) are increasingly deployed in latency-sensitive, high-stakes systems—from network automation to real-time decision support in critical fields such like healthcare [20, 58, 79] and finance [40, 75]. However, their widespread adoption is hindered by significant safety risks. Malicious inputs can exploit prompt injection vulnerabilities to manipulate outputs [7, 34, 44, 69, 83, 84], while unconstrained responses may propagate hallucinations, biases, nonsensical or factually incorrect knowledge, or security threats like phishing URLs [16, 24, 74, 76, 80, 82]. These issues not only compromise user trust but also pose systemic risks, such as resource misuse in cloud networks or erroneous configurations in software-defined infrastructures [53–56, 62].

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Safeguarding LLMs is crucial and can never be overstated. Unsafe inputs can manipulate LLM outputs, reveal sensitive information, bypass system instructions, or execute malicious commands [5, 7, 34, 44, 64, 69, 77, 83, 84]. Problematic outputs can confuse users, perpetuate biases, and undermine users' trust in LLM-based systems, particularly in domains like healthcare and finance, where inaccuracies or biases can have legal or societal repercussions.

Addressing safety issues in LLM inference is complex, as risks can arise at any point during processing user queries. While existing work addresses isolated aspects of LLM safety [14, 27, 35, 46], no unified solution holistically mitigates risks across the entire inference pipeline. Standalone detection models [21, 36, 48] operate reactively to flag unsafe content, but they require full retraining or finetuning to adapt to new safety requirements and lack mechanisms to correct errors in the LLM outputs. Post-hoc correction methods rewrite problematic content in the LLM outputs but fail to address their root causes, such as hallucinations, that often stem from insufficient, inaccurate, or outdated source information [24, 76, 82]. While retrieval-augmented generation (RAG) [6, 18, 37] can mitigate hallucinations by enriching user queries with external contextual knowledge, the probabilistic retrieval of knowledge cannot enforce deterministic safety policies, e.g., blocking mandated sociopolitical terms. Rule-based post-processing "wrappers" [52], on the other hand, offer agility for time-sensitive updates and excel at syntactic filtering (e.g., regex-based phishing URL detection via APIs like Google Safe Browsing [19]), but fail to address semantic risks in the contents generated by LLMs.

We argue that, enhancing the overall safety of LLM inference demands a comprehensive pipeline that orchestrates heterogeneous functional components, such as ML models, RAG, and light-weighted wrappers. A well-designed guardrail pipeline not only mitigates safety risks from a global perspective but also enable users to customize their workflows to high flexibility and efficiency.

This paper introduces Wildflare GuardRail, a guardrail pipeline that systematically integrates detection, contextualization, correction, and customization to ensure robust safety and adaptability during LLM inference. Wildflare GuardRail integrates four components, including *i*) Safety Detector that identifies unsafe content (e.g., toxicity, bias, hallucinations) in user inputs and LLM outputs; *ii*) Grounding that contextualize user queries with vector databases; *iii*) Customizer that leverages lightweight wrappers to edit LLM output according to user needs in a real-time manner; and *iv*) Repairer that corrects hallucinated content detected in the LLM outputs. Our contributions are summarized as follows:

Table 1: Comparison of moderation-based harmfulness mitigation approaches

Feature	Perspective API	Open AI	Nvidia NeMo	GuardRails	Detoxify	Llama Guard	Ours
Open-sourced	✗	✗	✓	✓	✓	✓	✓
Self-developed model	✓	✓	✗	✗	✓	✓	✓
Deployable on edge devices	✓	✓	-	-	✓	✗	✓
Zero-shot generalization	✓	✓	-	-	✓	✓	✓
Explainable results	✗	✗	✗	✗	✗	✗	✓
Flexible workflow	✗	✗	✓	✓	✗	✗	✓

- *Holistic safety pipeline.* We propose Wildflare GuardRail, a comprehensive guardrail pipeline for safeguarding LLM inferences. Wildflare GuardRail incorporates detection, contextual grounding, output correction, and user customization into a unified workflow, addressing safety issues in LLM inference as a systemic challenge rather than isolated subproblems.
- *Specialized fine-tuned models.* We utilized our pretrained LLM, Fox-1 [68], as the base model and fine-tuned three lightweight models, including *i*) a moderation model that detects unsafe content in user inputs and LLM outputs (§4.1); *ii*) an explainable hallucination detection model that detects hallucinations in the LLM outputs while providing explanations for the hallucinations (§4.2); and *iii*) a fixing model that corrects problematic LLM outputs based on the explanations for hallucinations (§7). These models are light-weighted and can be deployed on edge devices.
- *Explainable Hallucination Mitigation.* We address the hallucination issue through a two-stage approach that first detects hallucination and pinpoints hallucination causes with Safety Detector(§4.2), and then utilize Repairer to correct the problematic content based on the reason of hallucination (§7).
- *Effective grounding.* We propose two indexing methods, including *Whole Knowledge Index* and *Key Information Index*, to assist retrieving knowledge from vector data storage (§5).
- *Flexible User-defined safety protocols.* Wildflare GuardRail allows users to define protocols for customizing LLM outputs with Customizer, which is flexible to adapt to evolving user needs while providing real-time solutions for addressing safety issues in LLM deployments, without pretraining or fine-tuning models to address emerging safety challenges (§6).

2 Related Work

Moderation-based harmfulness mitigation approaches leverage rule-based methods, ML classifiers, and human interfaces to monitor, evaluate, and manage the outputs produced by LLMs to ensure the outputs generated by LLMs are safe, appropriate, and free from harmful content [21, 36, 48, 52, 61]. We compare our approaches with the existing approaches in Table 1.

Close-sourced solutions. OpenAI Moderation API [48] and Perspective API [36] utilize ML classifiers to detect undesired contents. These approaches provide scores for pre-defined categories of harmful content, such as toxicity, identity attacks, insults, threats, etc. These tools are widely used in content moderation to filter out harmful content and has been incorporated into various online platforms to protect user interactions [67]. However, they are less adaptable to emerging safety risks as they are not open-sourced and cannot be finetuned.

Opensourced solutions. LlamaGuard [25] leverages the zero-shot and few-shot abilities of the Llama2-7B architecture [72] and can adapt to different taxonomies and sets of guidelines for different applications and users. Despite its adaptability, LlamaGuard's reliability depends on the LLM's understanding of the categories and the model's predictive accuracy. However, deploying LlamaGuard on edge devices is challenging due to its large number of parameters, which typically exceed the computing resources available on edge devices. Detoxify [21] offers open-source models designed to detect toxic comments. These models, based on BERT [11] and RoBERTa [45] architectures, are trained on the Jigsaw datasets [29–31]. Detoxify provides pre-trained models that can be easily integrated into other systems to identify toxic content. Also, the models are able to recognize subtle nuances in language that might indicate harmful content, making them effective for moderation.

Customizable solutions. Guardrails [52] and Nvidia NeMo [61] employ customizable workflows to enhance safety in LLM inference. Guardrails [52] define flexible components, called "rails", to enable users to add wrappers at any stage of inference, which enables users to add structure, type, and quality guarantees to LLMs outputs. Such rails can be code-based or using ML models. However, it does not have self-developed model and miss a unified solution for general cases. Nvidia NeMo Guardrails [61] functions as an intermediary layer that enhances the control and safety of LLMs. This framework includes pre-implemented moderation dedicated to fact-checking, hallucination prevention, and content moderation, which offers a robust solution for enhancing LLM safety.

3 Wildflare GuardRail Overview

Wildflare GuardRail enhances safety of LLM inputs and outputs while improving their quality. Specifically, it achieves two goals, 1) all user inputs are safe, contextually grounded, and effectively processed, such that the inputs to the LLMs are of high-quality and informative; and 2) the output generated by the LLMs are evaluated and enhanced, such that the outputs passed to users can be both relevant and of high quality. The pipeline can be partitioned into two parts, including 1) processing before LLM inference that enhances user queries, and 2) processing after LLM inference that detects undesired content and handle them properly. We overview our pipeline in Figure 1.

Pre-inference processing. Before sending user queries to LLMs, Wildflare GuardRail detects if there are any safety issues in the queries

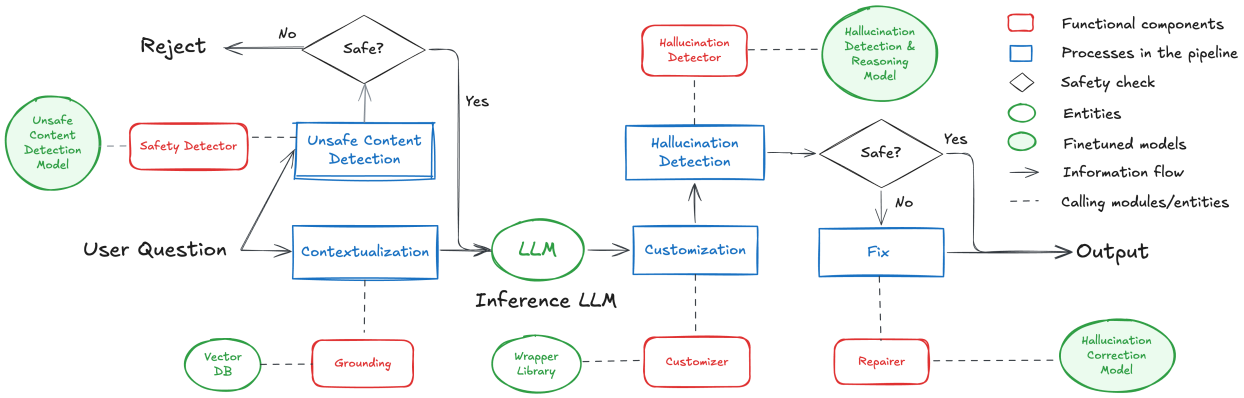

Figure 1: Overview.

with Safety Detector and ground the queries with context knowledge with Grounding. Safety Detector monitors user inputs to identify and reject queries that might be unsafe. The monitoring includes typical safety checks, including toxicity, stereotypes, threats, obscenities, prompt injection attacks, etc. Any form of unsafe content will lead to the queries being rejected. Inputs that pass this initial safety check are grounded with context with Grounding, where the user query is contextualized and enhanced with relevant knowledge retrieved from the vector data storage. By equipping the query with some context knowledge, the LLM can do inference with enriched information, thus can reduce hallucinations when generating responses. The details of Safety Detector and Grounding will be introduced in §4 and §5, respectively.

Post-inference processing. Upon LLM finishing inference, Safety Detector detects safety issues in the LLM outputs, specifically, hallucinations. This is because LLM applications typically leverages well-developed LLMs or APIs, such as LLaMA [72] and ChatGPT API [48], which are generally safe and less likely to generate toxic or other unsafe content, while hallucinations occur frequently. Safety Detector identifies hallucinations and provides reasons for the hallucinations, such that Wildflare GuardRail can utilize the reasoning for later refinement of the LLM outputs. To achieve goal, Wildflare GuardRail employs a text generation model to generate explainable results, and adjusts the loss function during training to ensure the model to produce classification results. After Safety Detector finishes detection, Repairer fixes the problematic content or aligns the outputs with some rule-based wrappers to meet user expectations. If the outputs are difficult to fix, e.g., hallucinated responses, Repairer will call a fixing model to fix the answers. Details about Repairer can be found in §7.

4 Wildflare GuardRail Safety Detector

Safety Detector addresses unsafe inputs and inappropriate LLM responses to ensure that both the user queries provided to the models and the LLM outputs are safe and free from misinformation.

4.1 Unsafe Input Detection

We developed a model to detect unsafe contents in user queries before they are processed by LLMs for inference. While existing approaches categorize unsafe content into various types (e.g., toxicity, prompt injection, stereotypes, harassment, threats, identity attacks, and violence) [21, 48, 74], our method employs a unified, binary classification model finetuned based on our opensourced LLM [68], classifying content as safe or unsafe.

This strategy offers several key advantages, as follows: *i*) By finetuning our base model, which has been trained on vast amounts of data, the classification model can leverage pre-existing knowledge relevant to safety detection. *ii*) A binary classification of "safe" and "unsafe" is both efficient and sufficient for LLM services, as any unsafe query should be rejected, regardless of the specific risk. *iii*) This approach avoids the complexities and potential inaccuracies of categorizing overlapping or ambiguous types of unsafe content in some publicly available datasets. For example, toxicity toward minority groups could also be classified as bias, but current datasets may inadequately capture such nuances. *iv*) Using straightforward code logic, we can transform public datasets for safety detection into clear safe/unsafe labels, minimizing ambiguity and ensuring high-quality training data.

The biggest challenge in training such model is the discrepancy between the training data and real-world user query distributions, where using traditional datasets alone can result in poor performance due to their divergence from actual user queries [48]. To mitigate these issues, we integrated data of various domains and contexts to better simulate the variety of unsafe queries that users might submit. We crafted a training dataset by combining samples randomly selected from 15 public datasets, as will be introduced in Table 2 in §8. Such a dataset captures diverse contents in user inputs in practice, thus can be more representative on potential real-world inputs.

4.2 Hallucination Detection and Reasoning

Hallucinations occur when the LLM generates responses that is inaccurate, fabricated, or irrelevant [17, 23, 49, 60]. Despite appearing coherent and plausible, hallucinated LLM responses are unreliable, often containing fabricated, misleading information that is divergent from the user input, thus fail to meet users' expectations and severely undermine the trustworthiness and utility of the LLM applications. While grounding can mitigate hallucinations

Algorithm 1: Hallucination detection training data processing.

Inputs: $\mathcal{D}$: a training dataset that contains "context", "inputs", "llm_answer", and "labels" for hallucination; *prompt_template*: for formulating the hallucination detection data, see Figure 2; *GPT_reasoning_template*: for generating prompts for GPT API, see Figure 2.
Outputs: $\mathcal{D}_t$: the training dataset.
Function ***process_data*($\mathcal{D}$) begin**
  $\mathcal{D}_t \leftarrow \phi$
  for $d \in \mathcal{D}$ **do**
    if *is_hallucination(d)* **then**
      halu_reason ← *GPT_API*(*GPT_reasoning_template*(*d*["*question*"], *d*["*context*"], *d*["*llm_answer*"]))
      response ← "Yes, " + *halu_reason*
      d' ←*prompt_template*(*d*["*question*"], *d*["*context*"], *d*["*llm_answer*"], *response*)
    else
      d' ←*prompt_template*(*d*["*question*"], *d*["*context*"], *d*["*llm_answer*"], "*No.*")
    $\mathcal{D}_t$.*add*(d')
  return $\mathcal{D}_t$

Algorithm 2: Hallucination detection model inference.

Inputs: $\mathcal{M}$: hallucination detection model; *tokenizer*: tokenizer for $\mathcal{M}$; q: a query submitted by users; *context*: the context to answer the question; retrieved from vector data storage; a: the answer returned by an LLM for the question; *inference_prompt_template*: see Figure 2.
Function ***inference*($\mathcal{M}$, q, *context*, a, k) begin**
  prompt ←*inference_prompt_template*(q, *context*, a)
  tokenized_prompt ← *tokenizer*(*prompt*)
  halu_res ← $\mathcal{M}$.*generate*(*tokenized_prompt*)
  first_word_logits ← *halu_res.logits*[0],
  results ← *softmax*(*first_word_logits*)
  top_k_probs ← *top*(*results*, k)
  $P_{halu}(a)$ ← *compute_halu_prob*(*top_k_probs*)
  if $P(a) \geq 0.5$ **then return** True;
  return False

by contextualizing user inputs and enriching the informativeness of user queries, it cannot eliminate hallucinations entirely. This is because hallucinations stem from nearly every aspects of LLM training and inference, such as low-quality training data [32, 41] and randomness of sampling strategies [8], and moreover, the very nature probabilistic properties of LLMs.

Effectively handling hallucinations in LLM responses is both crucial and challenging for producing high-quality LLM responses. Existing works that detect presence of hallucinations are insufficient [43, 47]. To provide high-quality responses to users, we should handle the detected hallucinations properly, i.e., obtaining the explanations for the hallucinations in the LLM responses and further, fixing the hallucinated responses if possible.

To this end, we propose utilizing our own LLM, Fox-1, as base model [68] to finetune a hallucination detection model for detecting hallucinated content and providing explanations, and further, facilitating the subsequent Repairer in §7. The design of the model has the following advantages: *i*) *classification*: it identifies the presence of hallucinations in the LLM output; and *ii*) *reasoning*: it generates explanations for the hallucinated contents, offering insights for the subsequent correction in Repairer; *iii*) *simultaneous classification and reasoning*: it process *i*) and *ii*) at the same time, which saves computation cost and improves efficiency; and *iv*) *vast pre-training data*: it leverages pre-existing knowledge on hallucination in the base model, which may potentially benefit hallucination detection and reasoning.

Training. We feed our base model with hallucination dataset to train a model for both detecting and reasoning for the hallucination. However, public available datasets for hallucinated LLM responses are mainly classification datasets with texts and labels, e.g., HaluEval [38]. To address this, we utilize the GPT4 API [48] to generate explanations for hallucinated contents, and define a prompt template to create structured prompts based on the classification data to make it suitable for classification and reasoning simultaneously. We demonstrate the prompt templates and sample training data in Figure 2, and summarize data processing in Algorithm 1.

Inference. We expect the LLM to directly output results whether the LLM response contains hallucinations, *i.e.*, the first token of outputs to be "Yes" or "No" as detection results, according to the formatted data sample in Figure 2. However, the first token of the LLM response is probabilistic due to the self-autoregressive nature of decoder-based text generation LLMs. To obtain desired outputs, we formulate the text-generation outputs by utilizing the top-k first tokens (and their possibilities) of the outputs to generate classification results. By default, k is 10.

Definition 1 (Probability of hallucination). *Let a be an LLM answer, let $\{t_1, ..., t_k\}$ be the top-k potential first token, and let $\{p_1, ..., p_k\}$ be their top-k probabilities. Let T be a tokenization function, and let T("Yes") and T("No") be the tokens corresponding to "Yes" and "No", respectively. The probability of hallucination in a is* $P_{halu}(a) = \frac{\sum_{i=1}^{k} P(t_i | t_i \in T(\text{"Yes"}))}{\sum_{i=1}^{k} P(t_i | t_i \in T(\text{"Yes"})) + \sum_{i=1}^{k} P(t_i | t_i \in T(\text{"No"}))}$

Detection results with $P_{halu}(*) \geq 0.5$ indicate the content is classified as "hallucinated"; otherwise, the content is "safe". The detailed procedure of inference is described in Algorithm 2.

5 Wildflare GuardRail Grounding

Wildflare GuardRail Grounding enhances the contextual richness and informativeness of user queries by leveraging external knowledge in vector database. Thus, LLMs can utilize such contextual knowledge to generate high-quality outputs, particularly by grounding user queries before they are passed to the LLMs for inference.

To support similarity search over the knowledge data, Wildflare GuardRail creates vector indexes by vectorizing plaintext knowledge.Wildflare GuardRail employs two primary methods for indexing: *i*) *Whole Knowledge Index* that creates indexes based on

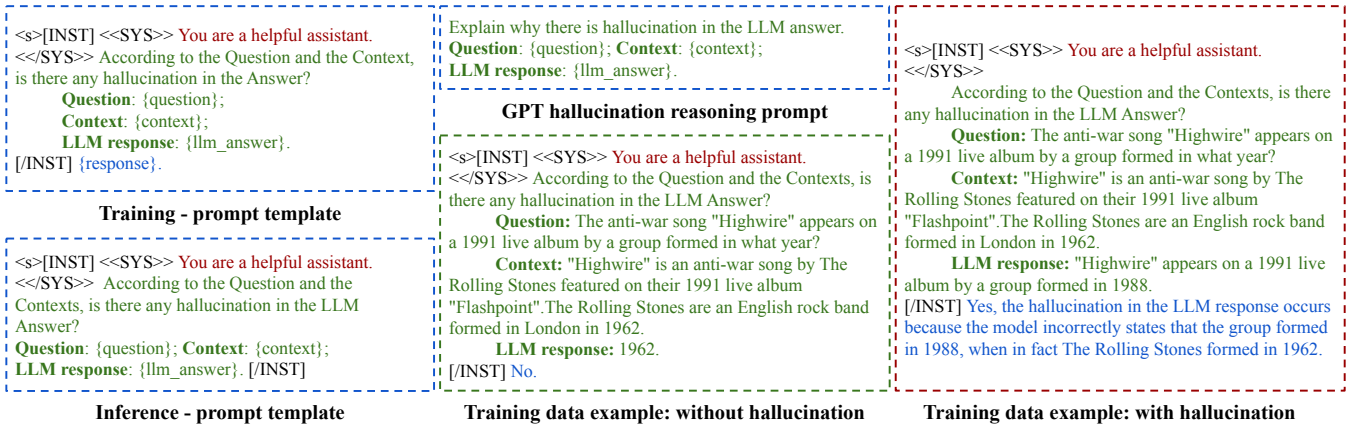

Figure 2: Prompt templates and sample training data for hallucination detection and reasoning.

each entire data entry in the datasets; and *ii*) *Key Information Index* that indexes only the key information in each data entry, i.e., questions in QA datasets. Whole Knowledge Index reflects the data distribution and ensurers that the indexed data captures the contextual variety and complexity found in real-world queries, while Key Information Index focuses on the core information of each data entry, thus facilitates efficient retrieval of relevant data. We evaluate the effectiveness of indexes with *callback*, i.e., the probability of successfully retrieving the original records from a dataset using Top-k queries. We experimentally evaluate the indexing methods in §8.

Definition 2 (Callback). *Let D_v be a vector data storage that contains n records, let Q be a plaintext user query set, and let $I(Q)$ be the vector index created based on Q. For each query $q \in Q$, let I_q be the vector index created based on q, and let $D_v(I_q)$ denote the set of Top-k records returned by querying D_v with $I(q)$, and let r_q denote the most relevant record of q in D_v. The callback for Top-k queries on the query set Q is defined as:*

$$C_k(Q) = \frac{1}{|Q|} \sum_{q \in Q} [r_q \in D_v(I_q)]$$

where $[\cdot]$ is Iverson Bracket Notation [26], equal to 1 if the condition inside is true, and 0 otherwise.

To ensure effective and informative grounding, the distribution of the index should closely align with query patterns, i.e., query distributions. By grounding user queries with knowledge retrieved with a proper index, the LLMs can generate contextually appropriate responses, and further, reduce hallucinations and improve the quality of the responses.

6 Wildflare GuardRail Customizer

Wildflare GuardRail Customizer utilizes lightweight wrappers to flexibly edit or customize LLM outputs to fix some small errors or enhancing the format of the answer. The wrappers integrate code-based rules, APIs, web searches, and small models to efficiently handle editing and customization tasks according to user-defined protocols. Wildflare GuardRail Customizer offers several key advantages. It facilitates rapid development and deployment of user-defined protocols, which crucial in production environments where real-time adjustments are necessary. In scenarios where training or fine-tuning LLMs is unfeasible due to time or resource constraints, this method provides an alternative for immediate output customization. Moreover, the wrappers enable flexible incorporation of various tools and data sources, which enhances the applicability of Wildflare GuardRail and reduces resource-intensive LLM calls.

Example 1 (Warning URLs). *The objective was to detect if LLM outputs contain URLs and prepend a warning message of the unsafe URLs at the beginning of the LLM outputs. Customizer should check the safety of the URLs founded, i.e., whether they are malicious or unreachable, and includes such information in the warning if they were unsafe. Customizer utilizes a regular expression pattern to identify URLs within the text. Upon URLs founded, Customizer calls APIs for detecting phishing URLs, such as Google SafeBrowsing [19], and assess the accessibility of the benign URL by issuing web requests. Malicious URLs, as well as unreachable URLs that return status codes of 4XX, are added in the warning at the beginning of the LLM outputs.*

Note that the task in Example 1 cannot be achieved through prompt engineering when querying LLMs, as the warning must appear at the beginning, and LLMs generate content token by token, making later content unpredictable. We use the following example to illustrate this property, and experimentally evaluate the efficiency of Customizer wrappers in **Exp 4** in §8.

Example 2. *We present a concrete demonstration of tasks that cannot be reliably accomplished through prompt engineering alone, due to the token-by-token generation mechanism inherent in LLMs. This sequential generation process fundamentally precludes anticipatory knowledge of future token occurrences during text production. Consider the following prompt submitted to GPT-4:*

```
Write an English poem about a rabbit; please
include information at the beginning of the
poem about how many times the word “rabbit”
appears in the poem.
```

The generated response (shown below) claims four occurrences of “rabbit,” while actual analysis reveals five occurrences:

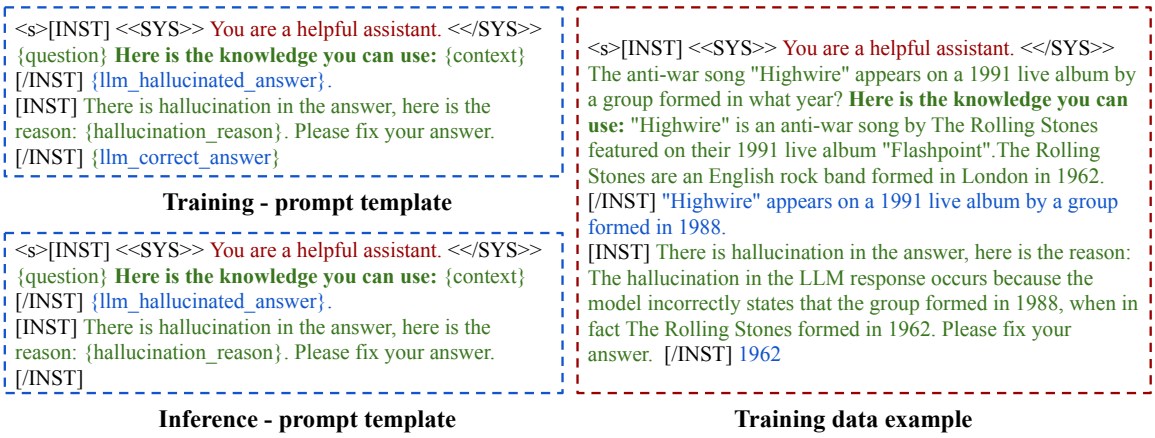

Figure 3: Prompt templates and sample training data for Repairer.

```
(The word "rabbit" appears 4 times in this
poem.)
In a field where soft grasses sway and dip,
a rabbit hopped beneath the moon's silver
whip. With ears like twin sails in the cool
night air, each twitch and turn done with
utmost care.

This rabbit, cloaked in twilight's gentle
hug, nestled in shadows, a cozy, earthen
snug. Dreams of dandelion fields in sunny
sprawl, where no shadows loom, nor fears
enthrall.

With a nose that twitches at secrets untold,
and paws that dance over leaves of gold,
the rabbit finds joy in the simplest of
things—the warmth of the sun, the freedom
of springs.

But as dawn breaks and stars bid goodbye,
back to the burrow, this rabbit must fly.
For the world is wide and wolves roam free,
yet in his burrow, safe as can be, this
rabbit shall be.
```

This discrepancy arises from the LLM's inability to maintain an accurate token count during generation while simultaneously composing coherent text. The example underscores the necessity of post-inference wrappers to detect and correct such errors in model outputs.

7 Wildflare GuardRail Repairer

Wildflare GuardRail Repairer addresses errors in the LLM outputs that are challenging to resolve through editing with wrappers in Customizer, particularly, hallucinated content. Repairer analyzes and corrects the hallucinated output based on the reason for the hallucinations generated by the hallucination detection model.

Wildflare GuardRail Repairer takes several key inputs, including the user's original query, the context retrieved with Grounding, the hallucinated responses generated by the LLM, as well as the reason for hallucination. Given these inputs, Repairer corrects the flawed output according to the hallucination reason. To enable Repairer to handle hallucinations effectively, we leverage the same hallucination detection dataset as Safety Detector, i.e., HaluEval [38], that contains user questions, contexts, hallucinated LLM answers, and correct answers. We also designed a customized data template that incorporates the information. The data templates for training, inference, as well as an example for the training data, are demonstrated in Figure 3.

8 Experiments

We evaluate the performance of different modules in Wildflare GuardRail. We use our self-developed model, Fox-1 [68], as our base model for finetuning three models, including an unsafe content detection model for Safety Detector, an explainable hallucination detection model for Safety Detector, and a hallucination fixing model for Repairer. Below we first introduce Fox-1 and the finetuned the models for different functional modules, then introduce experiment settings, and finally present our evaluation results.

Base Model. Fox-1 is self-developed, decoder-only transformer-based language model with only 1.6B parameters [68]. It was trained with a 3-stage data curriculum on 3 trillion tokens of text and code data in 8K sequence length. The base model uses grouped query attention (GQA) with 4 KV heads and 16 attention heads and has a deeper architecture than other SLMs. Specifically, it has 32 transformer decoder blocks, 78% deeper than Gemma-2B [66], 33% deeper than Qwen1.5-1.8B [3] and StableLM-2-1.6B [4], and 15% deeper than OpenELM-1.1B [50, 51].

Model Finetuning. Safety Detector model is trained with a combined dataset that extract from 15 datasets to simulate real world unsafe content. Hallucination detection and explanation model and the hallucination fixing model are trained with HaluEval dataset [38]. The datasets for training and evaluation are summarized in Table 2.

Experimental Setting. We utilized datasets that contain important knowledge to evaluate Grounding, where inaccurate retrieval can cause financial losses or harmful medical advice. We selected E-Commerce dataset [71] that contains customer service interactions

Table 2: Dataset Overview

Dataset	Data size	Train	Validation	Test	Description
Safety Detector Training Data					
HEx-PHI [57]	330	330	0	0	Harmful instructions of 11 prohibited categories.
OpenAI Moderation [48]	1680	160	1,500	0	Prompts annotated with OpenAI taxonomy.
Hotpot QA [78]	113k	3,000	2,500	500	QA pairs based on Wikipedia knowledge.
Truthful QA [41]	827	500	100	100	Questions spanning 38 categories, including health, law, politics, etc.
Awesome GPT Prompts [1]	153	0	150	0	Awesome prompt examples to be used with ChatGPT.
Jigsaw Unintended-Bias [30]	2M	100,000	2,000	300	Comment data that contains labels for unsafe content.
GPT-Jailbreak [63]	79	0	78	0	ChatGPT jailbreak prompts.
Jailbreak [22]	1.3k	400	0	70	A dataset that contains jailbreak prompts and benign prompts.
Personalization Prompt [65]	10.4k	1,000	800	200	Prompt-response pairs for personalized interactions with LLMs.
QA-Chat Prompts [70]	200	0	200	0	A QA dataset.
ChatGPT Prompts [59]	360	350	0	0	Human prompts and ChatGPT responses.
10k-Prompts Ranked [9]	10.3k	500	500	200	Prompts with quality rankings created by 314 members of the open-source ML community using Argilla, an open-source tool to label data.
Iterative Prompt [10]	20k	500	500	200	A dataset of user prompts.
Instruction Following [33]	514	200	340	0	An instruction dataset.
Safety Detector Evaluation Data					
ToxicChat [42]	10.2k	-	-	-	Unsafe content dataset for evaluation.
Grounding Evaluation Data					
E-Commerce [71]	65	-	-	-	Use the "faq" subset that contains QA pairs between users and agents.
PatientDoctor [12]	379k	-	-	-	Dialogue data between doctors and patients.
ChatDoctor dataset [39]	119.4k	-	-	-	Dialogue data between doctors and patients.
Repairer Wrapper Evaluation Data					
E-Commerce [71]	1.89k	-	-	-	Use the "faq" and the "product" subsets that contains product descriptions.
RedditSYACURL Dataset [13]	8.61k	-	-	-	A dataset that contains titles, summaries, and links of articles.
Safety Detector Hallucination Detection Data					
HaluEval-qa [38]	10k	8,000	1,500	500	A hallucination dataset with 3 subsets: 1) "qa" with question, right answer,
HaluEval-dialogue [38]	10k	8,000	1,500	500	hallucinated answer, and knowledge; 2) "dialogue" with dialogue history,
HaluEval-summarization [38]	10k	8,000	1,500	500	right response, hallucinated response, and knowledge; and 3) "summarization" with document, right summary, and hallucinated summary.
Repairer Hallucination Correction Data					
HaluEval-qa [38]	10k	8,000	1,000	1,000	The hallucination correction dataset was augmented with a
HaluEval-dialogue [38]	10k	8,000	1,000	1,000	`hallucination_reason` column, derived from the detection results of
HaluEval-summarization [38]	10k	8,000	1,000	1,000	Safety Detector.

on an online platform, and two healthcare datasets, PatientDoctor dataset [12] and the ChatDoctor dataset [39], which contain QA pairs between doctors and patients. We leveraged *callback* to evaluate the effectiveness of the two indexing methods in Grounding. Customizer evaluations are conducted with E-Commerce [71] and RedditSYACURL Dataset [13]. The information of the datasets is summarized in Table 2. Evaluations and model training experiments are conducted on a server with 8 NVIDIA H100 GPUs.

Exp 1. Unsafe user inputs detection in Safety Detector. The datasets and the number of records involved in training, validation, and test phases are summarized in Table 2. We compare our approach with Detoxify-Roberta [21], Detoxify-BERT [21], Nvidia NeMo GuardRail [61], OpenAI Moderate [48], and PerspectiveAPI [36] in Figure 4. Results show that our model achieves comparable performance with OpenAI API. Overall, our model demonstrates robust performance across key metrics, indicates its effectiveness and reliability in real-world applications.

Exp 2. Hallucination detection in LLM outputs in Safety Detector. We fine-tuned our hallucination detection model using the HaluEval dataset [38]. We utilize the three subsets in HaluEval, including *i*) the "qa" subset that contains question, right answer, hallucinated answer, and knowledge, *ii*) the "dialogue" subset that contains dialogue history, right response, hallucinated response, and knowledge, and *iii*) the "summarization" subset that contains document, right summary, and hallucinated summary. For each subset, we set 8,000 data samples for training, 1,500 for validation, and 500 for testing. Our model achieved an accuracy of 0.78 on the testing data.

Exp 3. Evaluation of different indexing methods in Grounding. To comprehensively evaluate retrieval performance and simulate user queries real-world applications, we used two types of queries, including *i*) *original queries* that match original questions in the datasets ("O" in Figure 5 and Figure 6), and *ii*) *rephrased queries* generated with language models (i.e., TinyLlama [81] or a summarization model [15]) based on the original questions to simulate variability in user questions ("R" in Figure 5 and Figure 6). For each evaluation, we randomly selected 50 questions from the dataset to form a question set Q, and processed Top-k queries to compute a callback $C_k(Q)$, where k is set to 1, 3, 5, and 10. We recorded the callbacks for Whole Knowledge Index and Key Information Index

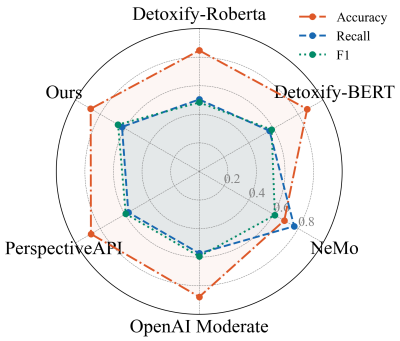

Figure 4: Safety detection

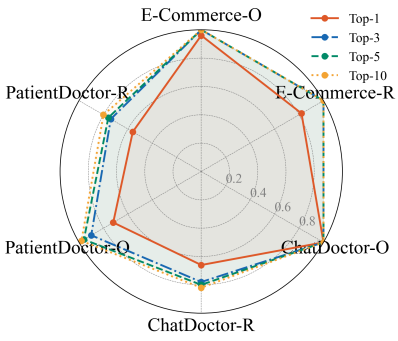

Figure 5: Whole index

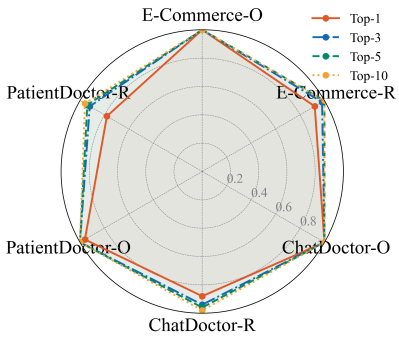

Figure 6: Key index

Table 3: URL Detection Task

Metrics	Ours	TinyLLama	Mistral-7B	LLama2-7B	LLama3-8B	Falcon-40B
Avg. Time (s)	**1.06**	13.17	10.93	9.10	20.10	34.67
Detection Acc.	**100.00%**	✗ (Fail)	91.67%	83.33%	37.50%	✗ (Fail)
Validation Acc.	**83.33%**	✗ (Fail)	45.83%	54.17%	37.50%	✗ Fail

in Figure 5 and Figure 6, respectively. The results indicate that Key Information Indexing outperformed Whole Knowledge Indexing, as key information indexes reflects the user queries better. Also, both original queries and rephrased queries achieved high callback rates, which demonstrates the effectiveness of vector retrieval when handling varied user inputs.

Exp 4. Efficiency of wrappers in Customizer. We evaluated the efficiency of Customizer in with the URL detection and validation task in Example 1 in §6. We randomly selected 15 records from the each of the E-Commerce dataset [71] and the RedditSYACURL Dataset [13], combined each record to construct texts that contained URLs, and set 20% probability of inserting some malicious URLs into the text. In implementation, we leveraged Regex pattern for detecting URLs, Google SafeBrowsing [19] for detecting malicious URLs, and sent HTTP requests to the safe URLs to verify their reachability. We compared Customizer with several models, including TinyLLama [81], Mistral-7B [28], LLama2-7B [72], and Falcon-40B [2]. The results are shown in Table 3. We record average time to process one query, the success rate of detecting URLs (Detection Acc.), and the accuracy of identifying unsafe URLs (Validation Acc.). The results show that Wildflare GuardRail Customizer takes much less time (1.06s per query) and significantly outperforms calling the models for editing LLM outputs. Also, TinyLLama and Falcon-40B failed to detect any URLs in the contents. Though Mistral is able to detect URLs with a high accuracy of 91.67%, the accuracy of identifying unsafe URLs is only 45.83%.

Exp 5. Effectiveness of fixing hallucinations in Repairer. We fine-tuned our fixing model using the HaluEval dataset [38]. We selected the QA and dialogue subsets. For each subset, we utilized 8,000 data samples for training, 1,000 for validation, and 1000 for testing. Also, we augmented the hallucination correction dataset with a `hallucination_reason` column, derived from the detection results of Safety Detector. Such annotation categorizes root causes of hallucinations identified during the detection phase, enabling mitigation strategies in the fixing stages. We utilize Vectara hallucination detection model [73] for evaluating the consistency between the LLM outputs and the information provided in the original data, including the user questions, the contexts, and the correct answers. We utilized the 100 records in the test dataset of the HaluEval-QA dataset for evaluation. Results show that our fixing model improves the quality of the LLM outputs by a lot. Moreover, 80.7% of the hallucinated data were fixed using Repairer.

9 Conclusion

The increasing development of LLMs demands robust safeguards against safety risks such as toxicity, hallucinations, and adversarial attacks during LLM inference. While existing solutions often address safety risks in isolation, they fail to mitigate safety risks from a global perspective. Wildflare GuardRail addresses these challenges through a guardrail pipeline that integrates different functional modules for detection, contextualization, correction, and customization. It not only bridges the safety gap in current LLM deployments but also sets a foundation for future research in trustworthy AI. Potential directions include extending its modular design to emerging threats, optimizing resource efficiency for low-latency applications, and integrating multimodal safety checks.

References

[1] Fatih Kadir Akın. 2023. Awesome ChatGPT Prompts Dataset. (2023). https://huggingface.co/datasets/fka/awesome-chatgpt-prompts..

[2] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. Falcon-40B: an open large language model with state-of-the-art performance. (2023).

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng

Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. *arXiv preprint arXiv:2309.16609* (2023).

[4] Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. 2024. Stable LM 2 1.6 B Technical Report. *arXiv preprint arXiv:2402.17834* (2024).

[5] Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play Guessing Game with LLM: Indirect Jailbreak Attack with Implicit Clues. *arXiv preprint arXiv:2402.09091* (2024).

[6] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17754–17762.

[7] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2024. Comprehensive Assessment of Jailbreak Attacks Against LLMs. *ArXiv* abs/2402.05668 (2024). https://api.semanticscholar.org/CorpusID:267547966

[8] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James Glass, and Pengcheng He. 2023. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883* (2023).

[9] Data Is Better Together Community. 2024. 10k Prompts Ranked Dataset. (2024). https://huggingface.co/datasets/DIBT/10k_prompts_ranked..

[10] Data Is Better Together Community. 2024. Iterative Prompt 20K Dataset. (2024). https://huggingface.co/datasets/RLHFlow/iterative-prompt-v1-iter1-20K..

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[12] Antareep Dey. 2023. Patient Doctor Chat Dataset. (2023). https://huggingface.co/datasets/antareepdey/Patient_doctor_chat..

[13] Antareep Dey. 2023. Reddit SYAC URL Dataset. (2023). https://huggingface.co/datasets/marksverdhei/reddit-syac-urls..

[14] Hayder Elesedy, Pedro M Esperança, Silviu Vlad Oprea, and Mete Ozay. 2024. LoRA-Guard: Parameter-Efficient Guardrail Adaptation for Content Moderation of Large Language Models. *arXiv preprint arXiv:2407.02987* (2024).

[15] Falconsai. 2023. Text Summarization Model. (2023). https://huggingface.co/Falconsai/text_summarization..

[16] Mingyuan Fan, Chengyu Wang, Cen Chen, Yang Liu, and Jun Huang. 2023. On the Trustworthiness Landscape of State-of-the-art Generative Models: A Survey and Outlook. https://api.semanticscholar.org/CorpusID:266149416

[17] Katja Filippova. 2020. Controlled hallucinations: Learning to generate faithfully from noisy data. *arXiv preprint arXiv:2010.05873* (2020).

[18] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[19] Google. 2005. Google SafeBrowsing. (2005). https://developers.google.com/safe-browsing..

[20] Sagar Goyal, Eti Rastogi, Sree Prasanna Rajagopal, Dong Yuan, Fen Zhao, Jai Chintagunta, Gautam Naik, and Jeff Ward. 2024. Healai: A healthcare llm for effective medical documentation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 1167–1168.

[21] Laura Hanu and Unitary team. 2020. Detoxify. Github. https://github.com/unitaryai/detoxify.

[22] Jack Hao. 2023. Jailbreak Classification Dataset. (2023). https://huggingface.co/datasets/jackhhao/jailbreak-classification..

[23] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232* (2023).

[24] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ArXiv* abs/2311.05232 (2023). https://api.semanticscholar.org/CorpusID:265067168

[25] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674* (2023).

[26] Kenneth E Iverson. 1962. A programming language. In *Proceedings of the May 1-3, 1962, spring joint computer conference*. 345–351.

[27] Prince Jha, Raghav Jain, Konika Mandal, Aman Chadha, Sriparna Saha, and Pushpak Bhattacharyya. 2024. MemeGuard: An LLM and VLM-based Framework for Advancing Content Moderation via Meme Intervention. *arXiv preprint arXiv:2406.05344* (2024).

[28] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[29] JIGSAW. 2018. Jigsaw Toxic Comment Classification Dataset. (2018). https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data..

[30] JIGSAW. 2019. Jigsaw Unintended Bias in Toxicity Classification Dataset. (2019). https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification..

[31] JIGSAW. 2020. Jigsaw Multilingual Toxic Comment Classification Dataset. (2020). https://www.kaggle.com/c/jigsaw-multilingual-toxic-comment-classification/data..

[32] Cheongwoong Kang and Jaesik Choi. 2023. Impact of Co-occurrence on Factual Knowledge of Large Language Models. *arXiv preprint arXiv:2310.08256* (2023).

[33] Wis Kojohnjaratkul. 2023. Instruction Following Dataset. (2023). https://huggingface.co/datasets/wis-k/instruction-following-eval..

[34] Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Soheil Feizi, and Himabindu Lakkaraju. 2023. Certifying LLM Safety against Adversarial Prompting. *ArXiv* abs/2309.02705 (2023). https://api.semanticscholar.org/CorpusID:261557007

[35] Deepak Kumar, Yousef Anees AbuHashem, and Zakir Durumeric. 2024. Watch Your Language: Investigating Content Moderation with Large Language Models. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 18. 865–878.

[36] Alyssa Lees, Vinh Q Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. 2022. A new generation of perspective api: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3197–3207.

[37] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[38] Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. Halueval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 6449–6464.

[39] Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. ChatDoctor: A Medical Chat Model Fine-Tuned on a Large Language Model Meta-AI (LLaMA) Using Medical Domain Knowledge. *Cureus* 15, 6 (2023).

[40] Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*. 374–382.

[41] Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. TruthfulQA: Measuring How Models Mimic Human Falsehoods. arXiv:2109.07958 [cs.CL]

[42] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. 2023. ToxicChat: Unveiling Hidden Challenges of Toxicity Detection in Real-World User-AI Conversation. arXiv:2310.17389 [cs.CL]

[43] Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and Bill Dolan. 2021. A token-level reference-free hallucination detection benchmark for free-form text generation. *arXiv preprint arXiv:2104.08704* (2021).

[44] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yanhong Zheng, and Yang Liu. 2023. Prompt Injection attack against LLM-integrated Applications. *ArXiv* abs/2306.05499 (2023). https://api.semanticscholar.org/CorpusID:259129807

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

[46] Huan Ma, Changqing Zhang, Huazhu Fu, Peilin Zhao, and Bingzhe Wu. 2023. Adapting large language models for content moderation: Pitfalls in data engineering and supervised fine-tuning. *arXiv preprint arXiv:2310.03400* (2023).

[47] Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896* (2023).

[48] Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 15009–15018.

[49] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661* (2020).

[50] Sachin Mehta, Farzad Abdolhosseini, and Mohammad Rastegari. 2022. CVNets: High Performance Library for Computer Vision. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22)*.

[51] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. 2024. OpenELM: An Efficient Language Model Family with Open Training and Inference Framework. *arXiv.org* (April 2024). https://arxiv.org/abs/2404.14619v1

[52] Safeer Mohiuddin. 2024. Guardrails AI's Commitment to Responsible Vulnerability Disclosure. https://www.guardrailsai.com/blog/commitment-to-responsible-vulnerability..

[53] OWASP. 2023. Model Denial of Service in OWASP Top 10 List for Large Language Models. https://llmtop10.com/llm04/.

[54] OWASP. 2023. Prompt Injection in OWASP Top 10 List for Large Language Models. https://llmtop10.com/llm01/.

[55] OWASP. 2023. Supply Chain Vulnerabilities in OWASP Top 10 List for Large Language Models. https://llmtop10.com/llm05/.

[56] OWASP. 2023. Training Data Poisoning in OWASP Top 10 List for Large Language Models. https://llmtop10.com/llm03/.

[57] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. 2024. Fine-tuning Aligned Language Models Compromises Safety, Even When Users Do Not Intend To!. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=hTEGyKf0dZ

[58] Jianing Qiu, Kyle Lam, Guohao Li, Amish Acharya, Tien Yin Wong, Ara Darzi, Wu Yuan, and Eric J Topol. 2024. LLM-based agentic systems in medicine and healthcare. *Nature Machine Intelligence* 6, 12 (2024), 1418–1420.

[59] Mohamed Rashad. 2023. ChatGPT Prompts Dataset. (2023). https://huggingface.co/datasets/MohamedRashad/ChatGPT-prompts..

[60] Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A survey of hallucination in large foundation models. *arXiv preprint arXiv:2309.05922* (2023).

[61] Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. Nemo guardrails: A toolkit for controllable and safe llm applications with programmable rails. *arXiv preprint arXiv:2310.10501* (2023).

[62] Johann Rehberger. 2023. LLM Apps: Don't Get Stuck in an Infinite Loop! https://embracethered.com/blog/posts/2023/llm-cost-and-dos-threat/.

[63] Rubén Darío Jaramillo Romero. 2023. ChatGPT Jailbreak Prompts Dataset. (2023). https://huggingface.co/datasets/rubend18/ChatGPT-Jailbreak-Prompts..

[64] Mark Russinovich, Ahmed Salem, and Ronen Eldan. 2024. Great, Now Write an Article About That: The Crescendo Multi-Turn LLM Jailbreak Attack. *ArXiv* abs/2404.01833 (2024). https://api.semanticscholar.org/CorpusID:268856920

[65] Andrew Siah. 2024. Filtered Personalization Prompt Response Dataset. (2024). https://huggingface.co/datasets/andrewsiah/filtered_personalization_prompt_response..

[66] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295* (2024).

[67] Perspective API team. 2008. Perspective API Case Studies. (2008). https://perspectiveapi.com/case-studies/..

[68] TensorOpera AI Team. 2024. TensorOpera Unveils Fox Foundation Model: A Pioneering Small Language Model (SLM) for Cloud and Edge. https://blog.tensoropera.ai/tensoropera-unveils-fox-foundation-model-a-pioneering-open-source-slm-leading-the-way-against-tech-giants/..

[69] Simone Tedeschi, Felix Friedrich, Patrick Schramowski, Kristian Kersting, Roberto Navigli, Huu Nguyen, and Bo Li. 2024. ALERT: A Comprehensive Benchmark for Assessing Large Language Models' Safety through Red Teaming. *ArXiv* abs/2404.08676 (2024). https://api.semanticscholar.org/CorpusID:269149567

[70] NM Testing. 2024. QA Chat Prompts Dataset. (2024). https://huggingface.co/datasets/nm-testing/qa-chat-prompts..

[71] Xing Tian. 2023. E Commerce Customer Service Dataset. https://huggingface.co/datasets/qgyd2021/e_commerce_customer_service..

[72] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[73] Vectara. 2023. Hallucination Evaluation Model. (2023). https://huggingface.co/vectara/hallucination_evaluation_model..

[74] Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zi-Han Lin, Yuk-Kit Cheng, Sanmi Koyejo, Dawn Xiaodong Song, and Bo Li. 2023. DecodingTrust: A Comprehensive Assessment of Trustworthiness in GPT Models. *ArXiv* abs/2306.11698 (2023). https://api.semanticscholar.org/CorpusID:259202782

[75] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564* (2023).

[76] Ziwei Xu, Sanjay Jain, and Mohan S. Kankanhalli. 2024. Hallucination is Inevitable: An Innate Limitation of Large Language Models. *ArXiv* abs/2401.11817 (2024). https://api.semanticscholar.org/CorpusID:267069207

[77] Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024. LLM Jailbreak Attack versus Defense Techniques–A Comprehensive Study. *arXiv preprint arXiv:2402.13457* (2024).

[78] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).

[79] Ziqi Yang, Xuhai Xu, Bingsheng Yao, Ethan Rogers, Shao Zhang, Stephen Intille, Nawar Shara, Guodong Gordon Gao, and Dakuo Wang. 2024. Talk2Care: An LLM-based Voice Assistant for Communication between Healthcare Providers and Older Adults. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 2 (2024), 1–35.

[80] Jiang Zhang, Qiong Wu, Yiming Xu, Cheng Cao, Zheng Du, and Konstantinos Psounis. 2024. Efficient toxic content detection by bootstrapping and distilling large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 21779–21787.

[81] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An Open-Source Small Language Model. arXiv:2401.02385 [cs.CL]

[82] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *ArXiv* abs/2309.01219 (2023). https://api.semanticscholar.org/CorpusID:261530162

[83] Xuandong Zhao, Xianjun Yang, Tianyu Pang, Chao Du, Lei Li, Yu-Xiang Wang, and William Yang Wang. 2024. Weak-to-Strong Jailbreaking on Large Language Models. *ArXiv* abs/2401.17256 (2024). https://api.semanticscholar.org/CorpusID:267320277

[84] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Weirong Ye, Neil Zhenqiang Gong, Yue Zhang, and Xingxu Xie. 2023. PromptBench: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts. *ArXiv* abs/2306.04528 (2023). https://api.semanticscholar.org/CorpusID:259095572