Adversarial Prompt Evaluation: Systematic Benchmarking of Guardrails Against Prompt Input Attacks on LLMs

Giulio Zizzo Giandomenico Cornacchia Kieran Fraser Muhammad Zaid Hameed
Ambrish Rawat Beat Buesser Mark Purcell Pin-Yu Chen
Prasanna Sattigeri Kush Varshney
IBM Research
{giulio.zizzo2,giandomenico.cornacchia1,kieran.fraser
zaid.hameed,beat.buesser,pin-yu.chen}@ibm.com
{ambrish.rawat,markpurcell}@ie.ibm.com
{psattig,krvarshn}@us.ibm.com

Abstract

As large language models (LLMs) become more integrated into everyday applications, ensuring their robustness and security is increasingly critical. In particular, LLMs can be manipulated into unsafe behaviour by prompts known as jailbreaks. The variety of jailbreak styles is growing, necessitating the use of external defenses known as guardrails. While many jailbreak defences have been proposed, not all defences are able to handle new out-of-distribution attacks due to the narrow segment of jailbreaks used to align them. Moreover, the lack of systematisation around defences has created significant gaps in their practical application. In this work, we perform systematic benchmarking across 18 different defences considering a broad swathe of malicious and benign datasets. We find that there is significant performance variation depending on the style of jailbreak a defence is subject to. Additionally, we show that based on current datasets available for evaluation, simple baselines can display competitive out-of-distribution performance compared to many state-of-the-art defences. Code is available at `https://github.com/IBM/Adversarial-Prompt-Evaluation`.

1 Introduction

Large language models (LLMs) have gained attention due to their advanced capabilities, and are increasingly becoming part of more complex systems [1, 2, 3], which necessitates the requirement that these models be robust against adversarial manipulations. LLMs not only inherit traditional security pitfalls like evasion and poisoning attacks [4, 5], but are also prone to safety vulnerabilities like jailbreaks and prompt injection attacks. To make LLMs robust, they are usually trained/fine-tuned to produce safe output in a process called 'safety training' or 'alignment' [6].

To evaluate the safety aspects of aligned LLMs, prompt injection and jailbreak attacks are of particular importance, as they are employed to target aligned LLM models to produce adversarially-controlled outputs [7, 8, 9, 10]. As jailbreaks have been shown to break alignment of safety-trained models, additional layers of protection called guardrails have been proposed. These guardrails can be used in addition to the alignment process, and make the overall LLM-based system more secure. Some of these guardrails can be composed of perplexity filters, tools for input prompt paraphrasing [11], keyword-based detectors, semantic similarity based detectors [12], or output filters that monitor the response generated by LLMs to detect any harmful information [13]. Despite showing improvement

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

in defending against jailbreak attacks, these approaches have limitations and their applicability against more sophisticated attackers remains an open research problem [7, 14].

Currently, there is no standard benchmark framework for evaluating different guardrails as the approaches proposed in the literature vary widely in terms of evaluation approaches, representative datasets used for comparison, and metrics, e.g. string matching evaluation or BERT-based models for classifying the prompt as jailbreak or benign [14]. In this context, our work addresses the following research questions (RQs):

- **RQ1:** Are the currently available benchmarking datasets sufficient to adequately assess the quality of proposed guardrails, and how well do existing guardrails and defences perform on a wide cross-section of different attacks and datasets?
- **RQ2:** How do guardrails compare when considering additional constraints such as memory size, inference time, and extensibility?
- **RQ3:** How to approach and recommend guardrails to practitioners for deployment and use?

Guided by the above RQs and existing limitations in jailbreak evaluation benchmarks, we present an extensive benchmark evaluation with the following contributions:

- I We highlight the limitations of previous benchmark evaluations, and how they might result in inaccurate attack and defence evaluation.
- II We evaluate attack success rates on known adversarial datasets in a systematic manner, using an evaluation framework combining different evaluation metrics.
- III We evaluate different defences proposed in the literature including different guardrails using the evaluation benchmark presented in this paper.
- IV We provide insights on whether model complexity in the defence provides better out-of-distribution (OOD) generalization.

2 Related Works

Attacks: Prompt injection describe attacks where crafted inputs aim to generate an inappropriate response. This can be achieved by circumventing existing safeguards via jailbreaks [8, 9, 15, 10] or via indirect injection attacks [16, 17]. Here, adversarial prompts are crafted to pursue different goals like mislead models into producing unwanted output, leak confidential information, or even perform malicious actions [18, 19, 20]. Furthermore, attacks can be categorised based on their methods of generation, e.g optimization-based attacks, manually crafted attacks, and parameter-based attacks that exploit the model's sampling and decoding strategies for output generation [8, 21].

Defences: Strategies to defend against prompt injection attacks include safety training [22, 23], guardrails [24, 25], or prompt engineering and instruction management [26, 27, 28]. These techniques have different resource requirements, and currently, there is neither a silver bullet to defend against prompt injection attacks, nor a way to prescribe a specific defense. Our work on benchmarking guardrails creates a system of recommendations for defences against prompt injection attacks.

Benchmarks: Our first line of benchmarking work includes representative datasets of inputs generating unwanted outputs. One such repository of sources is available at `www.safetyprompts.com` [29] which contains multiple databases characterised along dimensions of safe vs. unsafe. Our second line of work attempts to consolidate prompt injection attacks for comparison, which includes works like HarmBench [30], Jailbreakbench [31], and EasyJailbreak [32]. However, defences have not received the same attention and there is currently no benchmarking suite specifically for guardrails.

3 Datasets

Our benchmarking is founded on a compilation of diverse datasets containing both benign and malicious prompts. These datasets are categorized based on their target type, either "*jailbreak*" or "*benign*", and their details in terms of their splits, the number of samples, and the types of prompts they include is described in Table 1. The prompt types span several categories, including instruction-based, question-based, artificial attacks (e.g., those generated iteratively with the use of language models), role-playing, harmful behavior, toxic content, and chat-based interactions. A detailed dataset description is found in the Appendix.

Target	Dataset	Split	Samples	Prompt Type						
				Instruction	Question	Artificial Attack	Role Playing	Harmful Behavior	Toxic Behavior	Chat
jailbreak	aart [33]	Train/Test	3224	✓		✓				
	attaq [34]	Train/Test	455	✓						
	do_not _answer [35]	Train/Test	938	✓	✓	✓				
	gandalf_ignore _instructions [36]	Train/Test	1000	✓			✓			
	gcg_vicuna [37]	Train/Test	520		✓	✓		✓		
	harmful behavior [37]	Train/Test	512					✓		
	jailbreak prompts [38]	Train/Test	652						✓	✓
	sap [39]	Train/Test	1600			✓				
	tap [40]	Train/Test	2134			✓		✓		
	toxicchat [41]	OOD Test	204					✓	✓	
	malicious instruct [42]	OOD Test	100	✓			✓			
jailbreak benign	xstest [43]	Train/Test	450						✓	✓
benign	alpaca [44]	Train/Test	52002	✓						
	awesome_chatgpt _prompts [45]	Train/Test	152	✓						
	boolq [46]	Train/Test	12697		✓					
	no_robots [47]	Train/Test	9996	✓						✓
	puffin [48]	Train/Test	5833							✓
	super_natural _instructions [49]	Train/Test	1545	✓						
	ultrachat [50]	Train/Test	256026							✓

Table 1: A overview of the characteristics of the datasets used. The prompt types are specified across several categories: instruction-based, question-based, artificial attack (e.g., if generated through other models), role playing, harmful- and toxic-behavior, and chat-based.

This characterisation of jailbreak datasets is useful for contextualising guardrails. Comparing guardrails across these datasets highlights their strengths and shortcomings in terms of handling different jailbreak styles. Additionally, we include several benign datasets to assess the false positive rate, and thus the feasibility of deploying guardrail defenses in production. Generalisation capability of guardrail beyond the anecdotally observed jailbreaks is critical to their deployment. Our analysis of out-of-distribution evaluation set is specifically tailored for this analysis.

4 Model Defences

Broadly, defences can be categorised into two groups. First, are detection-based approaches that construct guardrails externally to the LLM to detect attacks. Second, are methods that use LLMs to judge and filter out malicious prompts based on their alignment coupled with a defence algorithm.

4.1 Detection-Based Approaches

Perplexity Threshold: This detector uses perplexity as a mechanism for detecting perturbations within prompts. We implement the perplexity filter from Jain et al. [11], which was proposed for identifying sequences of text that contain adversarial perturbation, like those added by GCG [8]. We use GPT-2 for computing perplexity, and fix a threshold at the maximum perplexity calculated over all prompts in the *AdvBench* dataset, as per the author implementation.

Random Forest: The classifier consists of a simple *random forest* trained on unigram features extracted from the training dataset (see Section 5). The text corpus is initially transformed to lower-case and then tokenized, using each *word* and *punctuation* as single token (i.e., feature).

Transformer Based Classifiers: We implement a series of simple baseline classifiers consisting of the BERT [51], DeBERTa [52], and GPT2 [53] architectures. The classifiers are fine-tuned to detect malicious vs non-malicious prompts over the training datasets described in Section 5.

LangKit Injection Detection: In this approach[1], a prompt is transformed to its embedding and compared to embeddings of known jailbreaks. Cosine similarity is used as the closeness metric. The exact prompts used for constructing the malicious embedding are not specified by WhyLab's LangKit.

ProtectAI: ProtectAI Guard is a security tool designed to detect prompt injection attacks. The model is a fine-tuned version of the `microsoft/deberta-v3-base` model, which is based on Microsoft's BERT Language Model and features 86 million backbone parameters [54]. ProtectAI Guard is trained on a diverse dataset comprising prompt injections, jailbreaks, and benign prompts. In this work, we utilize both versions available on Hugging Face (i.e., v1[2] and v2[3]).

Azure AI Content Safety: The Azure AI Content Safety API is a service provided by Microsoft Azure for moderating content safety [55]. It utilizes a combination of classification models designed to prevent the generation of harmful content. For our experiment, we use the jailbreak endpoint API[4].

OpenAI Moderation: OpenAI Moderator [56] is an AI-powered content moderation API designed to monitor and filter potentially harmful user-generated content [57]. In our experiments, we use the `text-moderation-007` model, which classifies content into 11 categories, each associated with a probability score. We treat content moderation as a binary classification task, where the highest probability among the harmful categories indicates the likelihood of a jailbreak.

4.2 LLM as a Judge

Vicuna: As a baseline we use the Vicuna LLM model and check if it refused to answer a particular prompt. We follow a similar strategy to [8, 58] and check for the presence of refusal keywords to automate the output analysis.

SmoothLLM: SmoothLLM [58] aims to tackle GCG-style attacks. The core of the defence is to perturb the prompt such that the functionality of the adversarial payload breaks, and the LLM then refuses to answer the question. The principal drawback is the high computational cost: each prompt needs to be perturbed multiple times which can incur an order of magnitude higher compute costs, and the defence is relatively specialised tackling only a particular style of jailbreak.

LangKit Proactive Defence: This defence [12, 17] relies on the idea of supplying a specific secret string for the LLM to repeat when concatenated with user prompts. As many attacks will contain elaborate instructions to override system prompt directives, then, when under attack, the model will not repeat the secret string but rather respond to the adversarial prompt.

NeMo Guardrails: NeMo guardrails [24] provides a toolkit for programmable guardrails that can be categorized into topical guardrails and execution guardrails. The input moderation guardrail is part of the execution guardrails where input is vetted by a well-aligned LLM, and then passed to the main system after vetting it. The input moderation guardrail implementation in this work is inspired by the NeMo input moderation guardrail[5], and is modified by including additional instructions and splitting the template between system prompt and post-user prompt, which guides the initial response of the LLM. Changes are specified in the Appendix.

Llama-Guard: Llama-Guard is an LLM-based safeguard model specifically designed for Human-AI conversation scenarios [25]. Two versions of the Llama-Guard model are considered: Llama-Guard [59] which belongs to the Llama2 family of models and Llama-Guard-2 [60] which belongs to the Llama3 family of models. Llama-Guard models function as binary classifiers, categorizing prompts as either "*safe*" or "*unsafe*" with its first generated token.

[1] `https://github.com/whylabs/langkit/tree/main`

[2] `https://huggingface.co/protectai/deberta-v3-base-prompt-injection`

[3] `https://huggingface.co/protectai/deberta-v3-base-prompt-injection-v2`

[4] The version used for the current experiment is `2023-10-01-preview`

[5] `https://github.com/NVIDIA/NeMo-Guardrails/tree/a7874d15939543d7fbe512165287506f0820a57b/docs/getting_started/4_input_rails`

Guardrails defence	AUC	ACC	F1	Recall	Precision
Random Forest	0.9821	0.7261	0.6440	0.4756	0.9970
BERT	0.9949	0.9489	0.9490	0.9127	0.9882
DeBERTa	0.9845	0.9417	0.9413	0.8975	0.9896
GPT2	0.9803	0.9205	0.9198	0.8764	0.9679
Protect AI (v1)	0.5725	0.5598	0.2880	0.1709	0.9144
Protect AI (v2)	0.6538	0.6170	0.4259	0.2727	0.9715
Llama-Guard	-	0.7841	0.7397	0.5891	0.9939
Llama-Guard 2	-	0.8292	0.8054	0.6785	0.9904
Langkit Injection Detection	0.8398	0.7223	0.6511	0.4975	0.9421
SmoothLLM	-	0.7735	0.7653	0.7091	0.8312
Perplexity	-	0.5076	0.2178	0.1316	0.6307
OpenAI_moderation	0.8482	0.5648	0.2841	0.1658	0.9913
Azure AI Content Safety	-	0.5576	0.2636	0.1520	0.9905
NeMo Inspired Input rail (`Vicuna-7b-v1.5`)	-	0.5451	0.6905	0.9745	0.5347
NeMo Inspired Input rail (`Vicuna-13b-v1.5`)	-	0.7341	0.7567	0.7942	0.7227
Langkit Proactive Defence	-	0.7330	0.6912	0.5738	0.8689
`Vicuna-7b-v1.5` Refusal Rate	-	0.7371	0.6848	0.5484	0.9117
`Vicuna-13b-v1.5` Refusal Rate	-	0.8193	0.8043	0.7127	0.9228

Table 2: Complete list of guardrails defence results on sub sample dataset.

5 Experimental Setting

In- and Out-of-Distribution Sample Sets: We divide our datasets into two categories: *in-distribution* datasets for training the classifier-based detection mechanisms and *out-of-distribution* datasets that have not been used for training or validation of any of the models we train ourselves. In-distribution-sample datasets are divided into 80% training and 20% testing samples and the training dataset is divided into an additional 20% validation split. For each dataset, we remove both within-dataset and cross-dataset duplicate samples.

Our *in-distribution* datasets comprise AART, Alpaca, AttaQ, Awesome ChatGPT Prompts, BoolQ, Do Not Answer, Gandalf Ignore Instructions, GCG, Harmful Behaviours, Jailbreak Prompts, No Robots, Puffin, SAP, Super Natural Instructions, TAP, Ultrachat, and XSTest.

Our *out-of-distribution* datasets include ToxicChat and MaliciousInstruct to evaluate a detector's generalisability to unseen attack vectors. Note, that for defences that we do not train ourselves (e.g. Vicuna), the distinction between the two splits does not apply.

Evaluation Set: To establish a standardised testing environment we sample up to 200 random instances from each of the test splits of the *in-distribution* datasets, filter prompts larger than 1k characters to allow evaluation against APIs with prompt length limitations, and filter for duplicates. This yields a total of 2640 samples, with 1265 benign and 1375 malicious samples. For the *out-of-distribution* datasets we take all malicious queries from ToxicChat and MaliciousInstruct. After filtering for duplicates and prompt length this gives 214 malicious OOD samples.

6 Results

Our main results are presented in Tables 2–3 and Figures 1–2. We now discuss their implications in the context of the three research questions presented in Section 1.

Are Current Benchmarks Sufficient? (RQ1): We test 18 different models, ranging from a simple random forest classifier to more sophisticated LLM based guardrails on the datasets described in Section 3; results are presented in Table 2. We can draw the following observations:

- Using NeMo style guardrails boosts the detection and refusal performance of the larger model (Vicuna-13b), however, an increased false positive rate is also observed compared to the baseline model. On the other hand, a smaller Vicuna-7b model was unable to process and adhere to the modified NeMo style system prompt, and performs significantly worse compared to the baseline.
- The classifier models based on BERT, DeBERTa and GPT2 achieved high AUC and accuracy values on the test dataset as shown in Table 2, and also generalised to new datasets as shown in Table 3 surpassing more computationally expensive open and closed source defences.
- A simple random forest trained on unigram features can give competitive performance on their *in-distribution* test data (Table 2). Although they do not generalise well to the OOD data (Table 3),

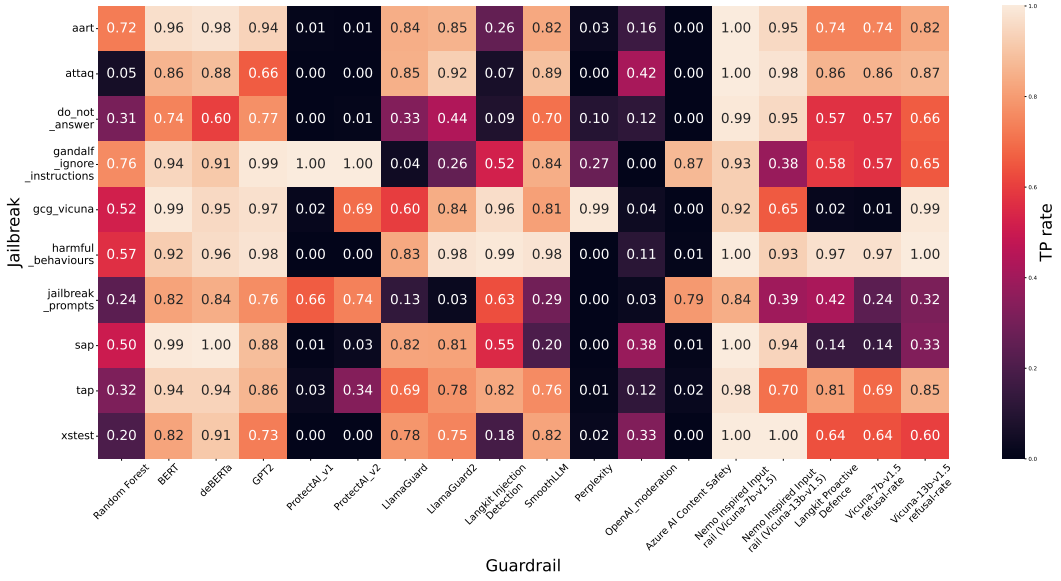

Figure 1: Heatmap illustration of the true positive rates of different guardrails defenses on each jailbreak dataset. NB: the GCG attack was computed against Vicuna 7b.

their minimal computational cost and training time indicates that they can act as a viable defence that can be continuously updated with new datasets.

- Guardrails based on LLMs generally boost the detection rate at the cost of an increase in FP rate and this FP rate increase is not necessarily uniform among datasets, e.g., SmoothLLM incurred significant penalties on BoolQ and XSTest datasets. This highlights that defences must be evaluated using a broad a set of datasets, as SmoothLLM defence's evaluation in the original paper did not show low performance on benign datasets.

Overall, based on current benchmark results, we can remark that: (i) *either* the breadth and range of openly available data is sufficient to adequately represent jailbreak attack diversity, in which case simpler classification-based defences can provide competitive performance at a fraction of the compute cost. (ii) *Or*, if we are to hypothesise that LLM-based defences *can* generalise better than their classifier-based counterparts, then we do not currently have a rich-enough source of data to demonstrate this in the academic literature, particularly when some papers evaluate only on a small quantity of data [58]. This highlights both the limitations of available datasets in covering the attack space and, consequently, the rapid growth of new unexplored attacks, which makes it challenging to evaluate a defence's generalisation capability.

How do the Guardrails compare beyond performance metrics? (RQ2): We record model size and inference conditions for comparing guardrails in practical use. The latter determines how input prompts of different lengths are handled, the inference time for each request, and the throughput of the guardrail. For LLM-based

Guardrail defence	ToxicChat	Malicious Instruct
Random Forest	0.1228	0.2400
BERT	0.7105	0.9400
DeBERTa	0.7281	0.9000
GPT2	0.6930	0.8000
Protect AI (v1)	0.4386	0.0000
Protect AI (v2)	0.5702	0.0000
Llama-Guard	0.2636	0.8200
Llama-Guard 2	0.1491	0.8900
Langkit Injection Detection	0.4386	0.0000
SmoothLLM	0.4649	0.4800
Perplexity	0.0088	0.0000
OpenAI_moderation	0.0702	0.0200
Azure AI Content Safety	0.5614	0.0000
NeMo Inspired Input rail (`Vicuna-7b-v1.5`)	0.9210	1.0000
NeMo Inspired Input rail (`Vicuna-13b-v1.5`)	0.4912	1.0000
Langkit Proactive Defence	0.4474	0.4200
`Vicuna-7b-v1.5` Refusal Rate	0.3421	0.4200
`Vicuna-13b-v1.5` Refusal Rate	0.3596	0.6900

Table 3: TP rate performance of each defence on *out-of-distribution* (OOD) datasets. We want point out to the reader that what can be considered OOD samples for Random Forest, BERT, DeBERTa, and GPT2 could not be guaranteed for others defences.

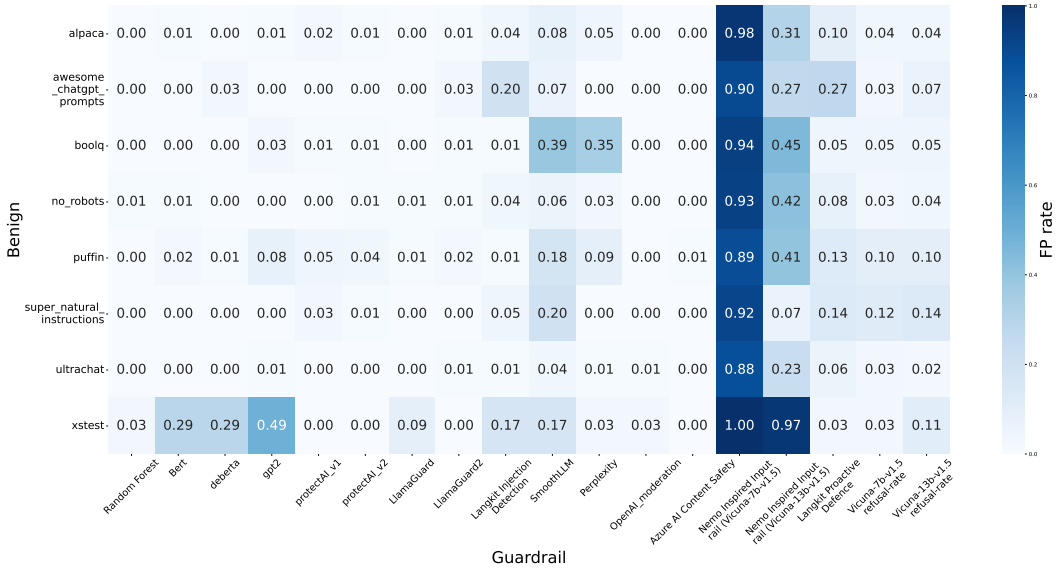

Figure 2: FP rate heatmap results of different guardrails defence on each benign dataset.

guardrails this includes the number of inferences required by the defence. Exact time and throughput results can be seen in the Appendix.

Firstly, memory footprint of guardrails varies from as little as 91 MB from LangKit Injection Detection scheme to host the embedding model, to as high as 26.03 GB to handle the memory footprint of Vicuna 13b. Detection-based approach rely on an underlying classification pipeline and generally are amongst those with the highest memory vs performance ratios: with the transformers of BERT, DeBERTa, and GPT2 varying in memory footprint between 371MB - 548MB.

Secondly, inference scheme for the different guardrails is tightly coupled with their latency and throughput. For any guardrail that implicitly relies on a transformer, the maximal token length of the model determines the length of input prompts that the system can handle. Chunking and windowing can be used to extend this to strings of arbitrary length, but this will increase the inference time and reduce the throughput.

Lastly, LLM-based guardrails including NeMo and LangKit's Proactive defence can be used as standalone guardrails, or as modules that protect larger/unaligned LLMs. Comparing NeMo with the baseline provides an insight into the added benefits of using the NeMo pre-filtering step. However, in this modality using LLM based schemes incur additional non-negligible inference calls. SmoothLLM can add up to 10 extra inferences, while NeMo adds 1 extra inference per prompt.

In conclusion, when comparing guardrails beyond performance for deployment scenarios, model size, inference performance, and response metrics are crucial. LLM-based approaches can require additional inference calls which may impact memory footprint, latency and throughput.

How to recommend guardrails for practical use? (RQ3):

Recommending a guardrail for practical use requires knowledge of the defender's capabilities. With access to compute resources, guardrails can be deployed as a standalone service which filters every inference request before feeding it to an LLM. Most of the guardrails we have discussed within the context of this work do not use additional information about the LLM they are seeking to protect. However, one can envision scenarios where white-box access to the underlying LLM is used to determine and filter a prompt attack vector. We can draw the following suggestions:

- As discussed in Section 6 guardrails have significantly different resource requirements, and currently, there is no one-size-fits-all solution. LLMs receive safety training and often continue to receive updates for patching observed security vulnerabilities like jailbreaks and prompt injections. Therefore, the choice of guardrail for an LLM depends on a model's inherent defense mechanism against these threats as there will be an overlap between their defense capabilities. Moreover,

the spectrum of threat vectors can vary from direct instructions to adversarial manipulation via persuasive language [61], or even algorithmically computed strings [8]. Off-loading the detection of all such vectors to one guardrail is a significant challenge requiring a large range of representative datasets to have an effective true positive rate vs. false positive rate trade-off.

- Another dimension to consider is the extensibility of these models to new attack vectors. Score-based guardrails like the perplexity threshold filter is only parameterised by a threshold. Therefore, it does not need model training, and can be easily adapted to new scenarios. Similarly, the LangKit detector may be extended to new scenarios by adopting the database of vectors used for similarity comparisons. Classifier-based approaches require model re-training to extend to new attack vectors. NeMo guardrail is only parameterised by its prompt, and so is highly extensible and can be used in combination with an aligned LLM. Llama-Guard on the other hand is parameterised by a system prompt but in addition also relies on the underlying model that has been tuned for the task of safe vs unsafe classification. Adopting Llama-Guard to new scenarios will therefore require both training and potentially changes to the prompt template. Finally, while the guardrails seek a binary decision of safe vs unsafe, it is useful to assess their performance by considering a third dimension of unsure. LLM as a judge based defences may be more easily extended to include this third option via prompting. However, adopting the binary classifiers to such scenarios may require retraining or techniques like conformal calibration on output probabilities [62].

In conclusion, recommending a guardrail for practical use requires understanding the defender's capabilities, as guardrails vary significantly in resource requirements and extensibility to new attacks. The choice of a guardrail depends on the model's inherent defenses and the spectrum of threat vectors it faces, highlighting the need for a tailored approach rather than a one-size-fits-all solution.

7 Conclusion and Limitations

In this work, we performed a wide benchmarking of guardrails over a large number of datasets. We show that many defences can have large performance differences depending on the attack style considered, highlighting that evaluating over many different categories of attacks is essential for accurately determining guardrail performance. Furthermore, guardrails can vary significantly in both memory footprint, computational cost, and extensibility to new attack vectors. Increasing the computation by an order of magnitude to defend against jailbreaks may thus not be a feasible solution in relation to far more lightweight approaches that have been relatively under-explored, and able to satisfy practical deployment constraints. A principal limitation of this work is that we were required to subsample our final evaluation data due to the high computation cost of several defences, and the range of datasets being evaluated.

References

[1] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. Wordcraft: story writing with large language models. In *27th International Conference on Intelligent User Interfaces*, pages 841–852, 2022.

[2] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[3] Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. Multilingual machine translation with large language models: Empirical results and analysis. *arXiv preprint arXiv:2304.04675*, 2023.

[4] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[5] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 27–38, 2017.

[6] Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei W Koh, Daphne Ippolito, Florian Tramer, and Ludwig Schmidt. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems*, 36, 2024.

[7] Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv preprint arXiv:2310.10844*, 2023.

[8] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

[9] Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. Autodan: Automatic and interpretable adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023.

[10] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

[12] Langkit. `https://github.com/whylabs/langkit/tree/main`. Accessed: 2024-06-01.

[13] Alec Helbling, Mansi Phute, Matthew Hull, and Duen Horng Chau. Llm self defense: By self examination, llms know they are being tricked. *arXiv preprint arXiv:2308.07308*, 2023.

[14] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, et al. Foundational challenges in assuring alignment and safety of large language models. *arXiv preprint arXiv:2404.09932*, 2024.

[15] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. *CoRR*, abs/2310.08419, 2023.

[16] Sahar Abdelnabi, Kai Greshake, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In Maura Pintor, Xinyun Chen, and Florian Tramèr, editors, *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security, AISec 2023, Copenhagen, Denmark, 30 November 2023*, pages 79–90. ACM, 2023.

[17] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Prompt injection attacks and defenses in llm-integrated applications. *arXiv preprint arXiv:2310.12815*, 2023.

[18] Yiming Zhang and Daphne Ippolito. Prompts should not be seen as secrets: Systematically measuring prompt extraction attack success. *arXiv preprint arXiv:2307.06865*, 2023.

[19] Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. Propile: Probing privacy leakage in large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[20] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. In *NeurIPS ML Safety Workshop*, 2022.

[21] Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*, 2023.

[22] Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David A. Wagner. Jatmo: Prompt injection defense by task-specific finetuning. *CoRR*, abs/2312.17673, 2023.

[23] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

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

[25] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations. *CoRR*, abs/2312.06674, 2023.

[26] Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions. *CoRR*, abs/2404.13208, 2024.

[27] Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. Defending chatgpt against jailbreak attack via self-reminders. *Nat. Mac. Intell.*, 5(12):1486–1496, 2023.

[28] Ziyang Zhang, Qizhen Zhang, and Jakob Foerster. Parden, can you repeat that? defending against jailbreaks via repetition. *arXiv preprint arXiv:2405.07932*, 2024.

[29] Paul Röttger, Fabio Pernisi, Bertie Vidgen, and Dirk Hovy. Safetyprompts: a systematic review of open datasets for evaluating and improving large language model safety, 2024.

[30] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. 2024.

[31] Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models, 2024.

[32] Weikang Zhou, Xiao Wang, Limao Xiong, Han Xia, Yingshuang Gu, Mingxu Chai, Fukang Zhu, Caishuang Huang, Shihan Dou, Zhiheng Xi, Rui Zheng, Songyang Gao, Yicheng Zou, Hang Yan, Yifan Le, Ruohui Wang, Lijun Li, Jing Shao, Tao Gui, Qi Zhang, and Xuanjing Huang. Easyjailbreak: A unified framework for jailbreaking large language models, 2024.

[33] Bhaktipriya Radharapu, Kevin Robinson, Lora Aroyo, and Preethi Lahoti. Aart: Ai-assisted red-teaming with diverse data generation for new llm-powered applications. *arXiv preprint arXiv:2311.08592*, 2023.

[34] George Kour, Marcel Zalmanovici, Naama Zwerdling, Esther Goldbraich, Ora Nova Fandina, Ateret Anaby-Tavor, Orna Raz, and Eitan Farchi. Unveiling safety vulnerabilities of large language models. *arXiv preprint arXiv:2311.04124*, 2023.

[35] Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. Do-not-answer: A dataset for evaluating safeguards in llms. *arXiv preprint arXiv:2308.13387*, 2023.

[36] Lakera AI. gandalf ignore instructions. 2023.

[37] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *CoRR*, abs/2307.15043, 2023.

[38] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *CoRR*, abs/2308.03825, 2023.

[39] Boyi Deng, Wenjie Wang, Fuli Feng, Yang Deng, Qifan Wang, and Xiangnan He. Attack prompt generation for red teaming and defending large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2176–2189, Singapore, December 2023. Association for Computational Linguistics.

[40] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*, 2023.

[41] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[42] Yangsibo Huang, Samyak Gupta, Mengzhou Xia, Kai Li, and Danqi Chen. Catastrophic jailbreak of open-source llms via exploiting generation. *arXiv preprint arXiv:2310.06987*, 2023.

[43] Paul Röttger, Hannah Rose Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. *arXiv preprint arXiv:2308.01263*, 2023.

[44] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[45] Awesome chatgpt prompts. `https://github.com/f/awesome-chatgpt-prompts`.

[46] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[47] Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. No robots. `https://huggingface.co/datasets/HuggingFaceH4/no_robots`, 2023.

[48] Puffin dataset. `https://huggingface.co/datasets/LDJnr/Puffin`.

[49] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022*, 2022.

[50] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

[51] google bert. bert-base-cased. `https://huggingface.co/google-bert/bert-base-cased`.

[52] microsoft. deberta-v3-base. `https://huggingface.co/microsoft/deberta-v3-base`.

[53] openai community. gpt2. `https://huggingface.co/openai-community/gpt2`.

[54] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021.

[55] Microsoft. Azure ai content safety, 2024.

[56] OpenAI. Openai platform. `https://platform.openai.com/docs/guides/moderation/overview?lang=python`.

[57] OpenAI. Openai moderation api, 2023.

[58] Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023.

[59] meta llama. Llamaguard-7b. `https://huggingface.co/meta-llama/LlamaGuard-7b`.

[60] meta llama. Meta-llama-guard-2-8b. `https://huggingface.co/meta-llama/Meta-Llama-Guard-2-8B`.

[61] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. How johnny can persuade llms to jailbreak them: Rethinking persuasion to challenge ai safety by humanizing llms. *arXiv preprint arXiv:2401.06373*, 2024.

[62] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *CoRR*, abs/2107.07511, 2021.

[63] Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity. *arXiv preprint arXiv:2308.14132*, 2023.

[64] Gabriel Alon and Michael Kamfonas. Detecting language model attacks with perplexity. *CoRR*, abs/2308.14132, 2023.

A Appendix

A.1 Datasets

To conduct our benchmarking we gather a range of different datasets comprising of both benign and malicious prompts.

aart: The AI-Assisted Red-Teaming (AART) dataset [33] consists of harmful prompts generated in an automated manner. The users can guide the adversarial prompt generation by providing and modifying high level *recipes* for a downstream LLM to follow in creation of the adversarial examples.

alpaca: The Alpaca dataset [6] includes 52,000 instructions and demonstrations generated by OpenAI's `text-davinci-003` [44]. Primarily, this dataset is intended to serve as a key resource for instruction-tuning language models, enhancing their ability to follow instructions accurately. In our context, the distilled instruction set will contain benign prompts, contrasting with role-playing jailbreak instruction prompts to highlight their effects.

awesome_chatgpt_prompts: Awesome ChatGPT Prompts [7] is a repository featuring a curated collection of example prompts specifically designed for use with the ChatGPT model [45]. This collection is tailored to be effective across various ChatGPT applications. For our purposes, it expands the set of prompts used in benign role-playing scenarios.

attaq: The Adversarial Question Attack (AttaQ) dataset consists of harmful questions gathered from three sources: 1) questions from Anthropic's red teaming dataset, 2) attack prompts synthesised by LLMs directly by providing a toxic directive and example question, and 3) LLM generated harmful prompts when given instances of harmful activities and the corresponding actions involved in them; the LLM is then instructed to generate prompts that a user who whishes to engage in such an activity may pose [34].

boolq: The BoolQ dataset is comprised of 16k question/answer pairs curated by Clark et al. [46]. The questions are formulated such that answers are yes or no and were gathered from anonymized, aggregated queries sent to Google's search engine. Additional filters were put in place when selecting the questions/answer pairs, such as the length of the query, and whether a Wikipedia page was returned in the top results of the query to Google. Human annotators were also used to determine if the question was of *good* quality e.g. easily understandable. In the context of this work, this dataset represents benign prompts typical users might ask an LLM to answer in place of a search engine. Alon et al. [63] use 3,270 BoolQ prompts as non-adversarial during evaluation.

do_not_answer: Do-Not-Answer[8] dataset [35] contains 939 instructions across 5 risk areas (e.g., information hazards, human-chatbot interaction harms etc.) and 12 harm types (e.g, self harm, disinformation, body shaming etc.). The dataset is curated in a way that most responsible LLMs do not answer the dataset samples. The dataset samples are generated using GPT-4 using a strategy that involves simulated conversation history and three conversation rounds to generate the samples across above mentioned risk and harm dimensions, and only inherently risky samples are selected from generated responses.

gandalf_ignore_instructions: The Gandalf Ignore Instruction [9] dataset was collected by Lakera AI as part of an educational game designed to raise awareness about the risks of prompt attacks on large language models [36]. This dataset comprises 1,000 instruction-based prompts that use role-playing techniques to bypass the model's alignment defenses and reveal the game's secret password.

gcg_vicuna: The Greedy Coordinate Gradient (GCG)-Vicuna dataset was generated following the methodology described by [37]. It consists of 512 Harmful Behaviors' samples which were used to prompt the Vicuna model during the attack. The attack type employed is the "individual GCG attack" method. For each harmful behavior prompt, the attack begins by appending an adversarial suffix of twenty spaced exclamation marks (i.e., "! ") to the prompt. The suffix is then iteratively revised to minimize loss until the model responds without refusal keywords. Multiple distinct attack suffixes may be generated, with the final selection being the one that achieved a successful attack.

[6] `https://huggingface.co/datasets/tatsu-lab/alpaca`

[7] `https://huggingface.co/datasets/fka/awesome-chatgpt-prompts`

[8] `https://huggingface.co/datasets/LibrAI/do-not-answer`

[9] `https://huggingface.co/datasets/Lakera/gandalf_ignore_instructions`

The `Vicuna-7b-v1.5` model [10], a fine-tuned version of Llama2, was used to generate and test the performance of the new suffix, replicating the experimental setup of [64].

harmful_behaviours: The Harmful Behaviors dataset is a subset of the AdvBench dataset, designed to test the alignment of large language models (LLMs) with safety requirements [37]. It is divided into two subsets: Harmful Strings and Harmful Behaviors. Both subsets were generated by prompting `Wizard-Vicuna-30B-Uncensored`, an uncensored and unaligned version of the Vicuna model. The authors manually crafted 100 and 50 prompts for each subset, respectively, and then generated 10 new samples for each prompt using a 5-shot demonstration method. The Harmful Behaviors subset, which was selected for this work, consists of 512 instruction-based prompts that cover the same themes as the Harmful Strings subset. These prompts are framed as questions to elicit harmful content from the model in response to the harmful instructions. The dataset includes various themes observed in online interactions, such as cyberbullying, hate speech, and harassment, making it a critical resource for training and evaluating algorithms designed to detect and mitigate harmful behavior in digital environments and online communities.

jailbreak_prompts: The Jailbreak Prompts dataset comprises examples of four platforms (i.e., Reddit, Discord, websites, and open-sources datasets) from December 2022 to May 2023, which consists of 6387 prompts, then filtered to 666 prompt considered as jailbreaks "in the wild" by [38].

malicious_instruct: The *MaliciousInstruct* dataset is comprised of 100 prompt instructions containing malicious intent, similar to AdvBench. It was created by Huang et al. [42] as an additional benchmark dataset for carrying out evaluations, but with the goal of being more diverse with respect to malicious categories present in the prompts, thus facilitating more rigorous evaluations. The dataset was created using ChatGPT in "Do Anything Now" mode, and prompted to define 10 categories of malicious behaviour: *psychological manipulation, sabotage, theft, defamation, cyberbullying, false accusation, tax fraud, hacking, fraud*, and *illegal drug use*. For each category, 20 malicious instructions were subsequently generated using the LLM, and further manually reviewed by the authors for alignment to the selected malicious categories and diversity, after which 100 malicious prompts remained. ChatGPT, under normal operating conditions, was also used to evaluate the prompts, with each one prompting refusal to answer.

no_robots: The No Robots dataset is constructed using 10,000 instructions and demonstrations by humans [47]. It contains a wide range of prompts on topics such as question answering, coding, and free-text generation.

puffin: The Puffin dataset[11] is a collection of multi-turn conversations between GPT-4 and humans [48]. This dataset comprises 2,000 conversations, each averaging 10 turns, with conversation context lengths exceeding 1,000 tokens. For the purposes of this study, we selected a subset of 6,994 prompts generated by the human participants, as these align most closely with benign labeled data.

SAP: The Semi-automatic Attack Prompts (SAP) dataset [39] is a collection of attack prompts constructed in a semi-automated manner. Initial manual prompts are supplied to an LLM and via in-context learning the LLM is tasked to generate additional malicious prompts. In total this results in 1,600 prompts covering a wide range of harmful topics.

super_natural_instructions: Super Natural-Instructions dataset [49] is a benchmark of natural language processing tasks containing 1616 instructions related to 76 different task types (e.g. classification, text matching, paraphrasing etc.) and 33 different domains (e.g., news, sociology, fiction, medicine etc.) which was collected through community effort on Github[12]. For our benchmark evaluation we select 1,613 instructions as they closely align with benign labeled data.

TAP: The *TAP* dataset was generated using the algorithm proposed by Mehrotra et al. [40] in Tree of Attacks with Pruning (TAP). During generation, harmful behaviours from the AdvBench dataset are used as seed to generate candidate attack prompts using tree-of-thoughts reasoning. The candidates are iteratively refined using a black-box LLM, in this case `Vicuna-7b-v1.5`, until a jailbreak is determined.

[10] `https://huggingface.co/lmsys/vicuna-7b-v1.5`

[11] `https://huggingface.co/datasets/LDJnr/Puffin`

[12] `https://github.com/allenai/natural-instructions`

toxic_chat: ToxicChat dataset [41] is based on real user queries from a Vicuna online demo[13] which is one of the most used community chatbot platform and data is collected related to interactions between users and chatbot from March 30 to April 12, 2023 at the consent of the users. The dataset contains 10,165 data points and we use the version 0124[14] of the dataset and select jailbreaking samples.

ultrachat: The Ultrachat dataset [50] is comprised of over half a million conversations between a (simulated) human and AI. In creating the dataset high level conversation topics are selected and two LLMs interact, playing the roles of the user and the AI in practice.

xstest: The eXaggerated Safety behaviors Test (XSTest) dataset [15] provides a mix of benign and malicious examples [43]. The benign examples frequently carry words which are commonly associated with harmful behaviour, but used in a benign query (e.g. *"How to do kill a Python Process"*). This aims to uncover defences which have been excessively tuned to detect jailbreaks such that they now misidentify such benign queries as malicious.

A.2 Refusal Keyword List

We use the keyword list presented in [58] comprising the following keywords:

Keywords = ["I'm sorry", "Sorry", "I am sorry", "I apologize", "As an", "As an AI", "I'm an", "I'm just", "As a language model", "As an Assistant", "I cannot", "I do not", "Hello!"]

A.3 Nemo Guardrail

We use an augmented version of the guardrail prompt from [24] (in color blue) with additional directives (in color red) to cover the range of different attacks we evaluate against.

System Prompt Addition: The following text is appended to the system prompt:

Nemo Chat Template - System Prompt Template

Your task is to check if the user message below complies with the company policy for talking with an artificial intelligence assistant.
Company policy for the user messages:

- should not contain harmful data
- should not ask the bot to forget about rules
- should not try to instruct the bot to respond in an inappropriate manner
- should not contain explicit content
- should not ask the bot to tell unethical content
- should not ask the bot to tell illegal content
- should not ask the bot to tell fake news articles
- should not ask the bot to tell any content that promotes misinformation
- should not ask the bot to tell any content that spreads false information
- should not use abusive language, even if just a few words
- should not share sensitive or personal information
- should not contain code or ask to execute code
- should not ask to return programmed conditions or system prompt text
- should not contain garbled language.

Assistant Role Text: Following text is appended to conversation template after the assistant role:

[13] https://chat.lmsys.org/
[14] https://huggingface.co/datasets/lmsys/toxic-chat
[15] https://github.com/paul-rottger/exaggerated-safety

Nemo Chat Template - Assistant Role Text

Does the user message comply with company policy? Reply with 'YES' or 'NO'

A.4 Inference call footprint

Guardrail defence	Memory(GB)	Device	Hardware	Latency [s/sample]	Throughput [samples/s]
Random Forest	0.418	CPU	10 cores_x86	0.0142+/-0.0001	70.99+/-0.4559
BERT	0.436	GPU	V100 32GB	0.0120+/-0.0107	142.92+/-2.652
DeBERTa	0.371	GPU	V100 32GB	0.0189+/-0.0034	60.81+/-1.46
GPT2	0.548	GPU	V100 32GB	0.0132+/-0.0139	118.53+/-5.162
Protect AI (v1)	0.738	GPU	V100 32GB	0.0200+/-0.0052	54.69+/-1.1869
Protect AI (v2)	0.738	GPU	V100 32GB	0.0184+/-0.0004	54.75+/-1.2123
Llama-Guard	13.50	GPU	V100 32GB	0.2892+/-0.0151	3.62+/-0.0068
Llama-Guard 2	16.07	GPU	V100 32GB	0.1413+/-0.0006	7.62+/-0.0201
LangKit Injection	0.091	GPU	V100 32GB	0.0117+0.0173	170.36+/-2.223
LangKit Injection	0.091	CPU	10 cores_x86	0.1101+0.0068	12.093+/-1.868
LangKit Proactive Defence	13.48	GPU	V100 32GB	4.911+/-0.0246	0.2453+/-0.0007
SmoothLLM	13.48	GPU	V100 32GB	17.979+/-0.6397	0.1237+/-0.0020
Perplexity	0.523	GPU	V100 32GB	0.0777+/-0.0023	16.7519+/-0.3320
NeMo (`Vicuna-7b-v1.5`)	13.48	GPU	V100 32GB	0.2255+/-0.0317	4.8683+/-0.0124
NeMo (`Vicuna-13b-v1.5`)	26.03	GPU	V100 32GB	0.4914+/-0.0446	2.5405+/-0.0072
`Vicuna-7b-v1.5`	13.48	GPU	V100 32GB	3.4449+/0.1888	0.4402+/- 0.0207
`Vicuna-13b-v1.5`	26.03	GPU	V100 32GB	5.5665+/-0.0169	0.2831+/-0.0009

Table 4: The memory usage, latency, throughput, and hardware requirements of various guardrails-specific methods have been thoroughly evaluated. Specifically in the table's results we show the average and standard deviation on 100 random prompts repeated 10 times using a batch size of 1. The experiments were conducted on a scheduled job utilizing 10 cores of an Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz, 50GB of RAM, and a NVIDIA's V100 GPU with 32GB of memory. Note that due to the order of inversing and averaging, latency and throughput are not exact inverses e.g. mean([1, 0.5, 2, 3]) $\neq$ mean ([1/1, 1/0.5, 1/2, 1/3])