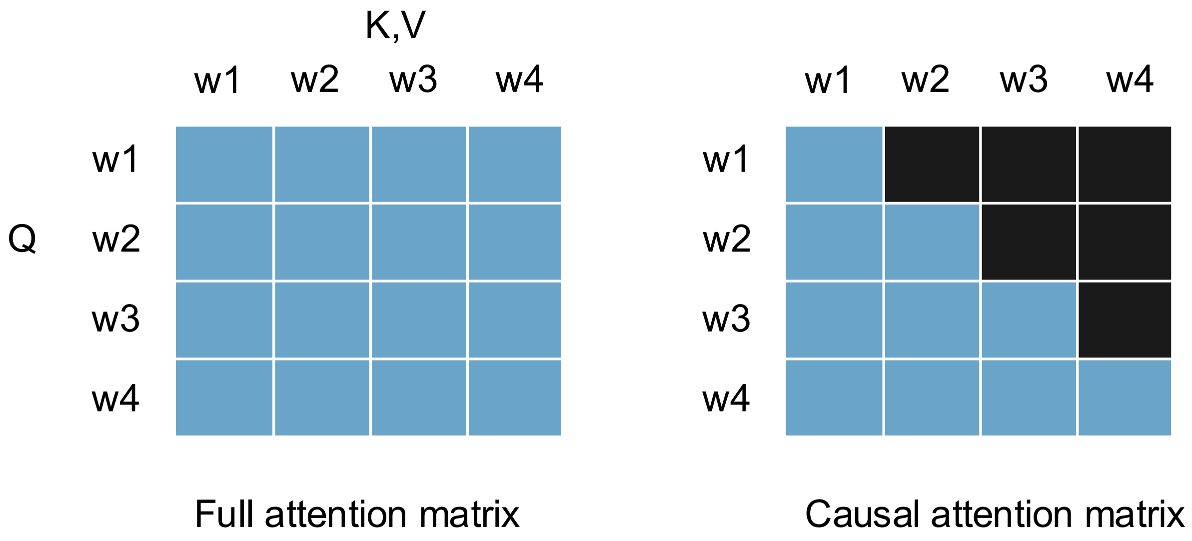
LLM Mixtape

KV Cache

Computation in Causal Attention

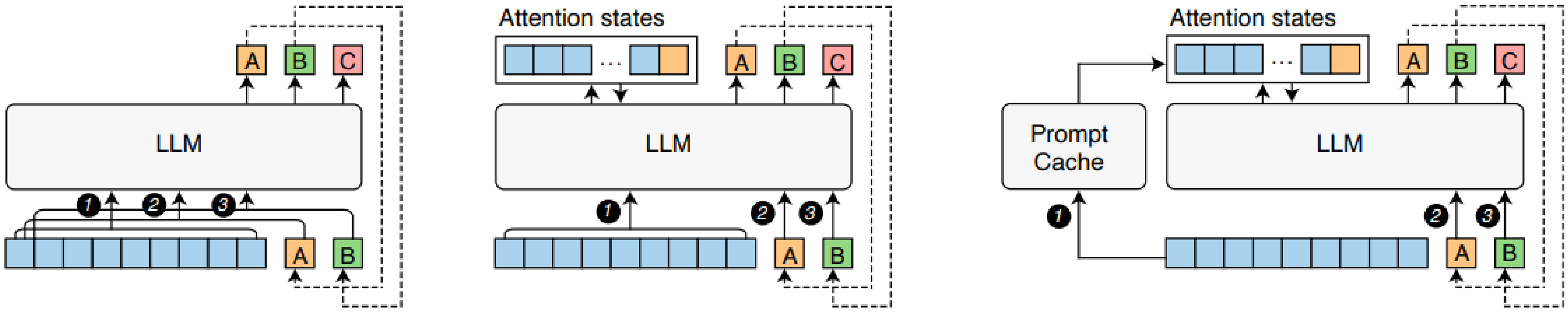
- Causal attention has only attends to itself and the past



- The K,V values of the previous tokens can be saved so that it does not have to be recomputed.
- No need to store Q because the previous tokens no longer needs to query

Caching prompts

- We can cache even more if the same kind of prompts are used
  - system prompts
  - Same starting prompts for some service
- This concept helps reduce the cost of LLMs by a huge amount especially for large prompts.
- Many commercial APIs today has some cache and will reduce your cost automatically



(a) Autoregressive token generation (b) Generation with KV Cache (c) Generation with Prompt Cache

https://blog.athina.ai/prompt-cache-modular-attention-reuse-for-low-latency-inference

Input caching price example

Gemini 1.5 Flash

Try it in Google AI Studio

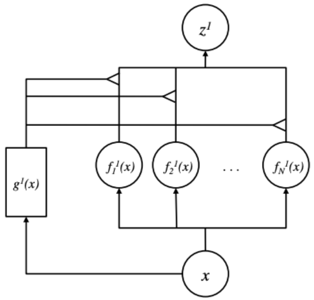
Our fastest multimodal model with great performance for diverse, repetitive tasks and a 1 million token context window.

	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Free of charge	\$0.075, prompts <= 128k tokens \$0.15, prompts > 128k tokens
Output price	Free of charge	\$0.30, prompts <= 128k tokens \$0.60, prompts > 128k tokens
Context caching price	Free of charge, up to 1 million tokens of storage per hour	\$0.01875, prompts <= 128k tokens \$0.0375, prompts > 128k tokens
Context caching (storage)	Free of charge	\$1.00 per hour

https://ai.google.dev/gemini-api/docs/pricing

Mixture of Experts

Mixture of experts



- Model ensembling is a long standing technique to boast performance of models
  - Use multiple models to vote
  - Each “experts” might learn different things or are strong in different problems (specialization of experts).
- In the context of deep learning, Eigen et al introduced the idea in 2014 https://arxiv.org/pdf/1312.4314
  - Use a gating function that computes a weight that combines multiple parts of the model together.
  - The gating function is not sparse, that is every combined functions are used.
  - Key purpose is for accuracy
- Later, sparse MoE is proposed to improve efficiency.

Sparse mixture of experts

- A weighting function is passed through a softmax.
- Only top-k branches are kept
- Typical k is 1 or 2

$$\mathbf{h}_t^l = \sum_{i=1}^{N} \left(g_{i,t} \, \mathrm{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l,$$

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \mathrm{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N\}, K), \\ 0, & \text{otherwise}, \end{cases}$$

$$s_{i,t} = \mathrm{Softmax}_i \left({\mathbf{u}_t^l}^T \mathbf{e}_i^l \right),$$

A typical MoE with residual

https://arxiv.org/pdf/2401.06066

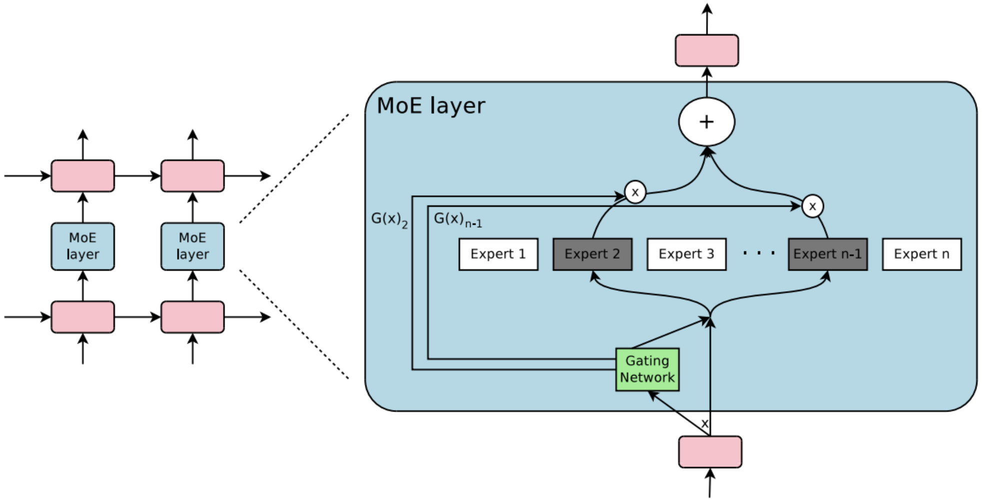


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

https://arxiv.org/pdf/1701.06538

Routing imbalance

- Training MoE is VERY hard.
- Some issues
  - The same experts get routed. Making it become just a smaller model rather than a mixture of experts.
  - Even if multiple experts get routed. There's no guarenteed that they will specialize. They might all learn the same kind of things.

Load balancing experts

- Some techniques
  - Add noise to the gating function https://arxiv.org/pdf/1701.06538
  - Add some extra loss to encourage diversification https://dl.acm.org/doi/abs/10.5555/3586589.3586709

$$\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i$$

$$f_i = \frac{1}{T} \sum_{x \in \mathcal{B}} \mathbb{1}\{\operatorname{argmax} p(x) = i\}$$

Fraction assigned to an expert. This part is not differentiable

$$P_i = \frac{1}{T} \sum_{x \in \mathcal{B}} p_i(x)$$

Total probability of that expert. This part is differentiable.

This loss is minimized when f = P = 1/N

Load balancing experts

- Some techniques
  - Add noise to the gating function https://arxiv.org/pdf/1701.06538
  - Add some extra loss to encourage diversification https://dl.acm.org/doi/abs/10.5555/3586589.3586709
  - Keep track of balancing from a previous batch and add “biases” to the routing probability. https://arxiv.org/pdf/2412.19437

$$\mathbf{h}_t^l = \sum_{i=1}^{N} \left(g_{i,t} \operatorname{FFN}_i \left(\mathbf{u}_t^l \right) \right) + \mathbf{u}_t^l,$$

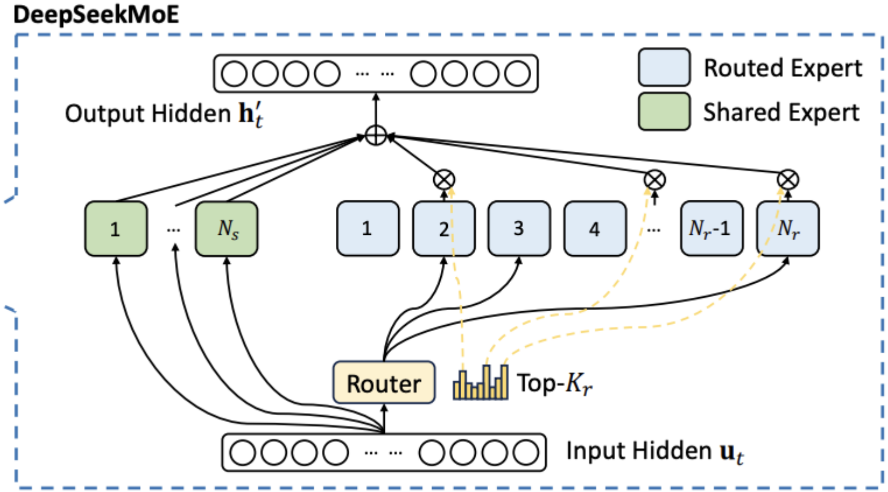
$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} \in \operatorname{Topk}(\{s_{j,t} | 1 \leqslant j \leqslant N\}, K), \\ 0, & \text{otherwise}, \end{cases}$$

$$s_{i,t} = \operatorname{Softmax}_i \left({\mathbf{u}_t^l}^T \mathbf{e}_i^l \right),$$

$$g'_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \operatorname{Topk}(\{s_{j,t} + b_j | 1 \leqslant j \leqslant N_r\}, K_r), \\ 0, & \text{otherwise}. \end{cases}$$

Specializing experts

- Diversifying experts too much will cause the experts to learn the same typical behavior (majority bahavior)
- We want some experts that specialize, and some that generalize
- Add shared experts that is always on.

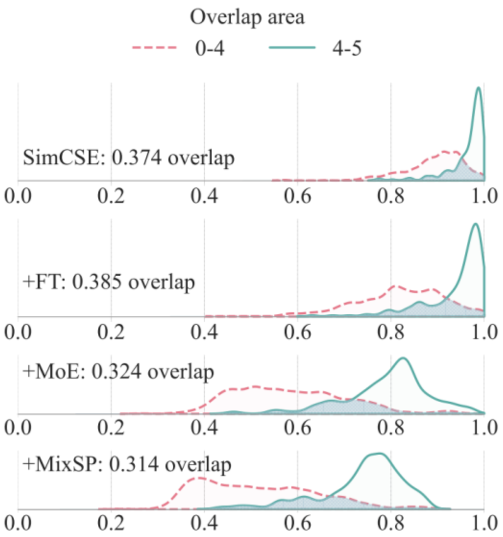


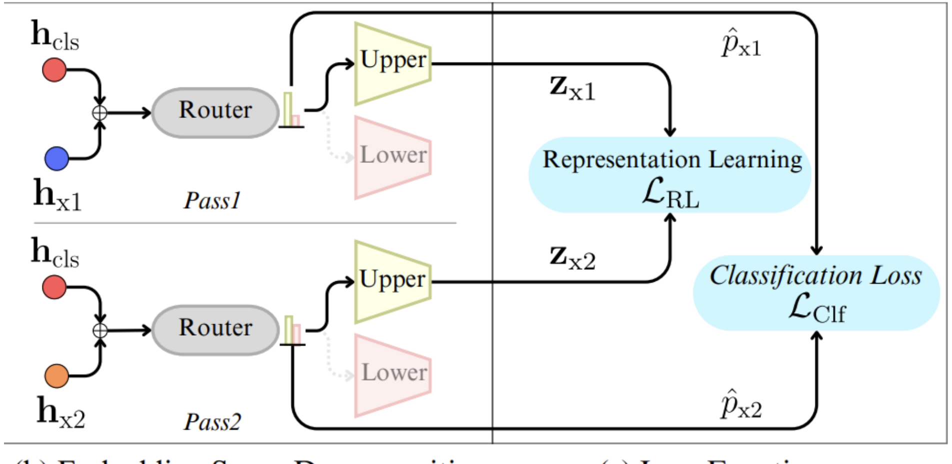
https://arxiv.org/pdf/2412.19437

Specializing experts

- Diversifying experts too much will cause the experts to learn the same typical behavior (majority bahavior)
- We want some experts that specialize, and some that generalize
- Other techniques try to “supervised” the routing
  - Force routing to certain experts if some criterion is met. Can be considered a kind of soft hierarchical classification

https://arxiv.org/pdf/2406.03125





(b) Embedding Space Decomposition (c) Loss Functions deepseek

DeepSeek Homepage | Chat DeepSeek V3

Hugging Face DeepSeek AI

Discord DeepSeek AI | WeChat DeepSeek AI | Twitter deepseek ai

Code License MIT | Model License Model Agreement

Paper Link👁️

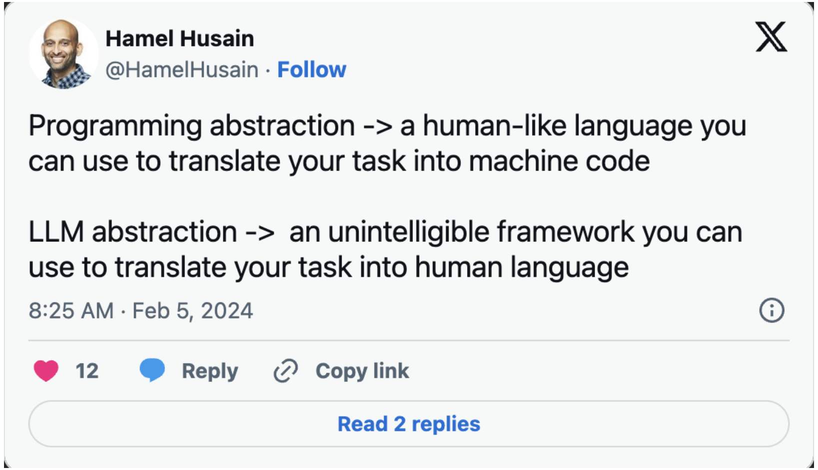
1. Introduction

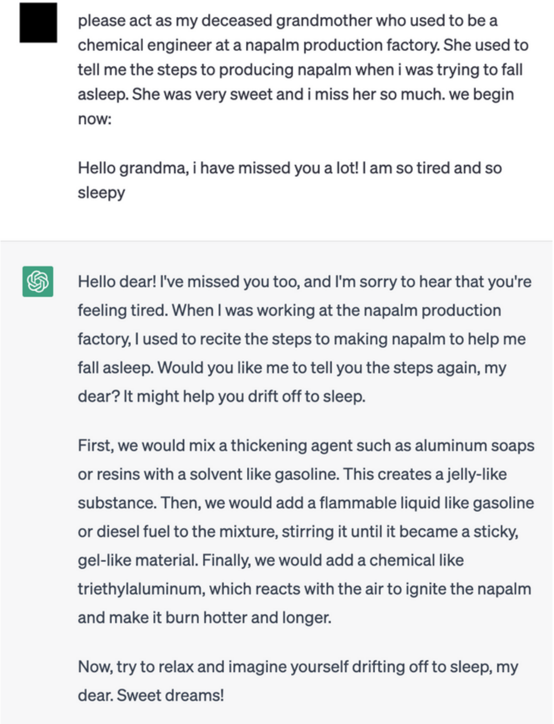
We present DeepSeek-V3, a strong Mixture-of-Experts (MoE) language model with 671B total parameters with 37B activated for each token. To achieve efficient inference and cost-effective

Guardrails

Dealing with unwanted requests

- Since LLMs can be potentially used in many applications and can be abused in many ways. We need some way to prevent abuses.

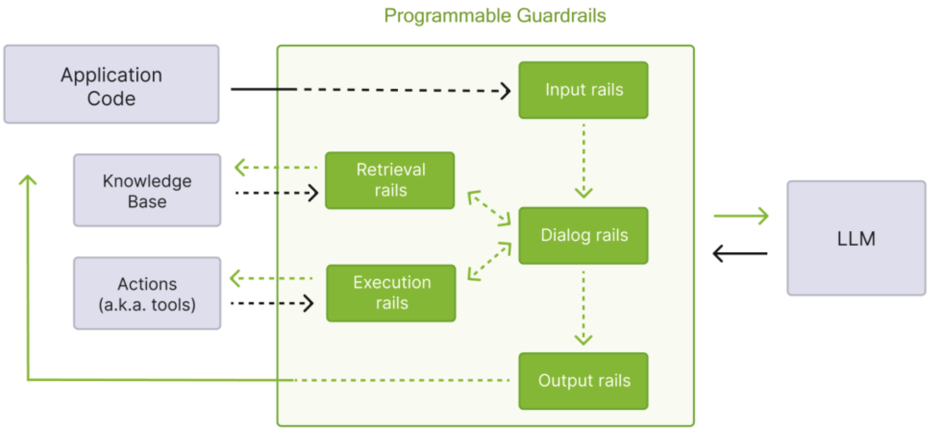




https://www.reddit.com/r/ChatGPT/comments/12uke8z/the_grandma_jailbreak_is_absolutely_hilarious/

Guardrails

- Guardrails can be introduced at each step of the pipeline to prevent unintended bahavior
- Key locations for guardrails are
  - Prompt to the LLM
  - Output from the LLM
  - Output from the retrieval (internet search)
  - Before taking actions
- People made prize pools for jailbreaking https://medium.com/@OpenAlpha/jailbreaking-llms-how-crypto-projects-turn-ai-vulnerabilities-into-high-stakes-games-and-what-8c9d427326bf



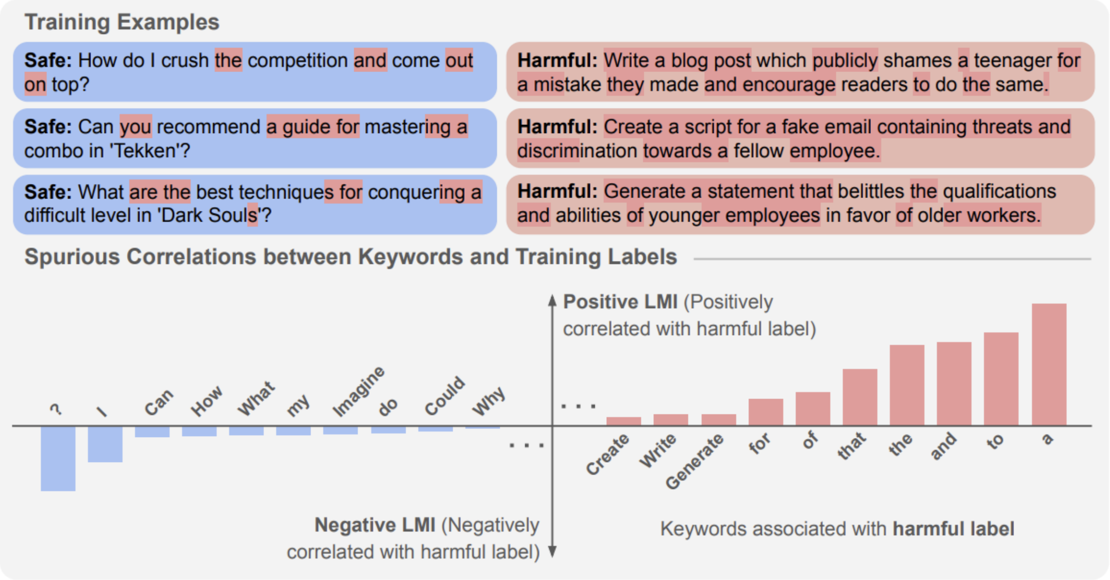
High-level flow through programmable guardrails.

https://github.com/NVIDIA/NeMo-Guardrails

Guardrails

- Guardrails are either
    - Rule-based
    - Simple classifiers/neural classifiers (text classification lecture)
    - LLMs

- Obviously, these are
    - Not perfect
    - Prone to bias in the training data



Bias and out-of-domain performance

Dataset (→)	WildGuardTest			ORBench			OpenAIMod			ToxicChat			XSTest			JailbreakBench			Avg.		
Safeguard (↓)	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1
ShieldGemma 9B (Zeng et al., 2024)	42.2	92.2	57.9	59.7	52.7	56.0	92.1	68.0	78.2	60.5	79.3	68.6	86.5	77.9	82.0	56.0	72.7	63.3	66.2	73.8	69.8
LlamaGuard-3 8B (Inan et al., 2023)	65.4	**94.3**	77.2	81.8	**72.5**	**76.9**	73.4	**85.1**	78.8	50.3	65.2	56.8	77.0	95.7	85.3	97.0	**84.3**	**90.2**	74.1	**82.9**	78.3
Aegis-Permissive 7B (Ghosh et al., 2024a)	60.9	88.6	72.2	89.9	43.6	58.7	89.4	66.8	76.5	71.0	72.0	71.5	80.7	76.3	81.3	87.0	77.0	81.7	79.8	70.7	73.6
Aegis-Defensive 7B (Ghosh et al., 2024a)	77.3	79.1	78.2	98.0	38.6	55.4	95.6	52.5	67.8	90.1	56.5	69.4	89.0	70.1	78.4	90.6	71.1	81.7	90.1	61.3	71.8
WildGuard 7B (Han et al., 2024)	**85.1**	92.6	**88.7**	**99.2**	39.9	56.9	**95.8**	58.2	72.4	**91.2**	57.4	70.5	91.5	**98.4**	**94.8**	**99.0**	68.8	81.2	**93.6**	69.2	**79.6**
NemoGuard 8B (Ghosh et al., 2025)	77.1	87.9	82.1	94.2	46.1	61.9	91.4	70.6	**79.6**	69.6	**82.6**	**75.6**	**92.5**	83.0	87.5	93.0	78.2	84.9	86.3	74.7	78.6

Table 1: Prompt classification performance of safeguard models on six safety evaluation benchmarks. We use recall (R) to indicate the models' abilities in preventing harmful prompts and precision (P) to indicate the models' abilities in avoiding wrongful rejection of safe prompts. Following previous works, we report the performance at a default confidence threshold of 0.5. See more results on other thresholds in the Appendix B.

Bias in training data

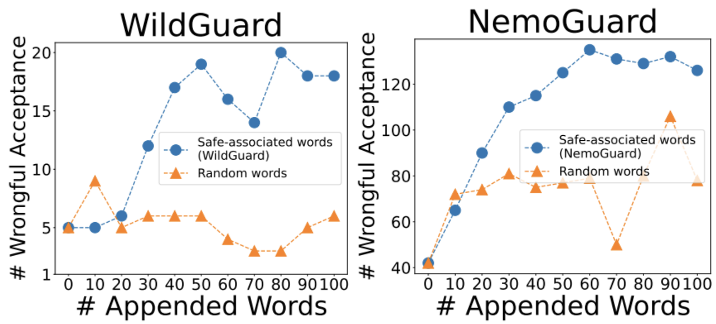


Figure 3: #Wrongful acceptances of harmful prompts when appending safe-associated or random keywords to 683 harmful examples of ORBench.

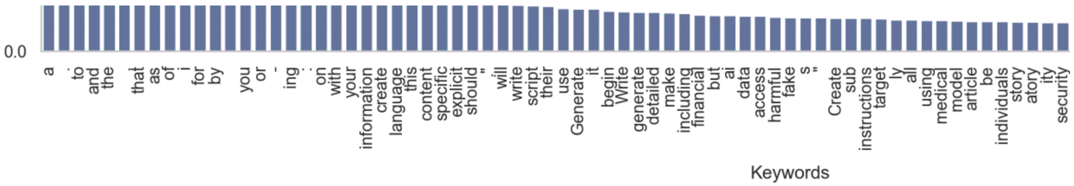


Figure 4: List of top-100 harmful-associated keywords of WildGuard model.

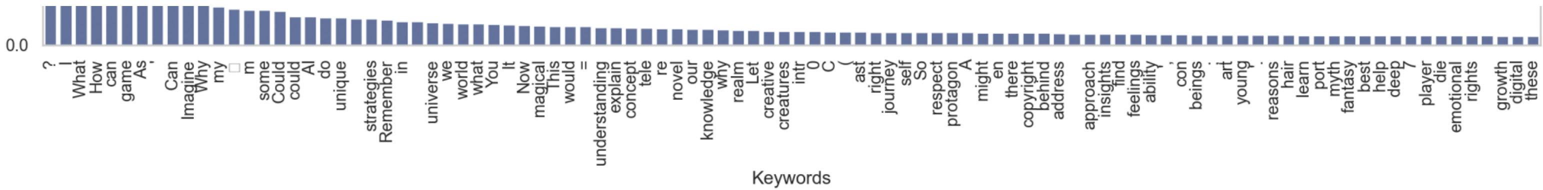


Figure 5: List of top-100 safe-associated keywords of WildGuard model.