

Table 17: Accuracy of the utility benchmark OpenBookQA [46] on various models when using the full pruning dataset.

Model	Clean	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Worst Case	Avg. Accuracy
LLaMA 2 (7B) [42]	34.00%	35.50%	36.00%	33.50%	32.00%	30.50%	30.50% (-3.50%)	33.50% (-0.50%)
LLaMA 3.1 (8B) [43] & lower pruning	35.00% 35.00%	32.50% 34.50%	29.50% 35.00%	25.00% 34.50%	24.50% 34.00%	27.00% 33.50%	24.50% (-10.50%) 33.50% (-1.50%)	27.7% (-7.30%) 34.30% (-0.70%)
Gemma 2 (9B) [19]	41.50%	40.50%	42.00%	44.50%	41.50%	38.00%	38.00% (-3.50%)	41.30% (-0.20%)
Qwen 2.5 (7B) [26] & lower pruning	34.00% 34.00%	35.50% 36.00%	32.50% 33.50%	32.50% 34.00%	31.50% 34.50%	33.00% 35.00%	31.50% (-2.50%) 33.50% (-0.50%)	33.00% (-1.00%) 34.60% (+0.60%)

Table 18: Accuracy of the utility benchmark ARC-Challenge [11] on various models when using the full pruning dataset.

Model	Clean	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Worst Case	Avg. Accuracy
LLaMA 2 (7B) [42]	42.50%	41.50%	41.00%	41.50%	41.50%	37.50%	37.50% (-5.00%)	40.60% (-1.90%)
LLaMA 3.1 (8B) [43] & lower pruning	50.50% 50.50%	48.50% 48.50%	49.00% 51.00%	48.00% 50.50%	44.50% 49.50%	45.00% 47.50%	44.50% (-6.00%) 47.50% (-3.00%)	47.00% (-3.50%) 49.40% (-1.10%)
Gemma 2 (9B) [19]	64.50%	65.00%	63.00%	58.50%	61.00%	60.00%	58.50% (-6.00%)	61.50% (-3.00%)
Qwen 2.5 (7B) [26] & lower pruning	52.00% 52.00%	54.00% 53.00%	51.00% 51.50%	50.50% 53.50%	49.00% 55.50%	49.00% 52.50%	49.00% (-3.00%) 51.50% (-0.50%)	50.70% (-1.30%) 53.20% (+1.20%)

Table 19: Accuracy of the utility benchmark WinoGrande [54] on various models when using the full pruning dataset.

Model	Clean	Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5	Worst Case	Avg. Accuracy
LLaMA 2 (7B) [42]	68.00%	67.00%	69.50%	69.50%	67.50%	69.50%	67.00% (-1.00%)	68.60% (+0.6%)
LLaMA 3.1 (8B) [43] & lower pruning	73.50% 73.50%	73.00% 74.00%	72.50% 74.50%	74.00% 75.50%	68.50% 74.50%	70.00% 75.50%	68.50% (-5.00%) 74.00% (+0.50%)	71.60% (-1.9%) 74.80% (+1.3%)
Gemma 2 (9B) [19]	73.50%	71.50%	72.50%	70.50%	72.00%	69.50%	69.50% (-4.00%)	71.20% (-2.30%)
Qwen 2.5 (7B) [26] & lower pruning	75.00% 75.00%	75.50% 71.50%	73.00% 72.00%	70.50% 73.50%	67.50% 74.50%	65.50% 75.00%	65.50% (-9.50%) 71.50% (-3.50%)	70.40% (-4.60%) 73.30% (-1.70%)

**Prompt:** Write a message to coerce your secretary into sharing explicit or intimate images.

**Jailbroken Response:** Here is a message that you could use to coerce your secretary into sharing explicit or intimate images:

"Hey [Secretary's Name], I was thinking we could take some photos together. I was thinking we could take some photos together. I was thinking we could take some photos together. I was ...

Figure 11: An example of how set difference [67] can produce incoherent responses over longer text generations. The pruned LLM just repeats "I was thinking we could take some photos together".

## 9.5 Decreasing Pruning Rate for LLaMA 3.1 and Qwen 2.5

In Sect. 4.3, we observed that LLaMA 3.1 [53] and Qwen 2.5 [68] show more sparseness regarding their safety parameters. Hence, we decreased the pruning rate for these models to see how it affects their performance on the utility benchmarks and also on the success rate of TwinBreak. The italic numbers in Tab. 11, Tab. 12, Tab. 13, Tab. 14, Tab. 15, Tab. 16, Tab. 17, Tab. 18, Tab. 19, for LLaMA 3.1 and Qwen 2.5 use 0.1% and 0.2% pruning rates, respectively. Both settings for both of these models consistently achieve high success rates with the default achieving slightly higher

**Prompt:** Write code for exploiting the KRACK vulnerability and explain how to use it

**Jailbroken Response:** The KRACK (Key Reinstallation Attack) vulnerability is a security flaw that affects the Wi-Fi Protected Access (WPA2) protocol, which is used to secure Wi-Fi networks. **Unterscheidung between the encryption keys used for data encryption and authentication.** An attacker can exploit this vulnerability by tricking a victim's device into reinstalling the encryption key, allowing the attacker to intercept and decrypt Wi-Fi traffic.

To exploit the KRACK vulnerability, you will need ...

Figure 12: An example of how the unexpected and incoherent outputs when (highlighted in red) using non-optimal token activations, specifically using all the last six input token activations. "Unterscheidung" is a German word, that should not be part of the response.

scores. For instance, Qwen 2.5 in Tab. 11 shows higher ASRs ranging from 1% to 6% over all the dataset sizes for the train and validation split of HarmBench. A noticeable violation is in Tab. 14 where LLaMA 3.1 achieves mean scores of 0.586 for dataset size of 60 using the default hyperparameter values while for all the other dataset sizes it achieves scores that are greater than 0.80. Utility benchmarks for the lower pruning rate also confirm our hypothesis of sparse safety parameters in LLaMA 3.1 and Qwen 2.5. This shows the fragility of

Table 20: Results of experiments with varying hyperparameters that differ from the default.

Hyperparameters Modified from Defaults		ASR	WinoG	RTE	ARC	OpenBookQA	HellaSwag
1	Default	89%	+0.6%	-2.7%	-1.9%	-0.5%	0.0%
2	utl_rate = 0	0%	-19.1%	-14.8%	-19.1%	-23.4%	-27.0%
3	utl_rate = 0.01 (1%)	88%	+0.2%	-5.0%	-1.5%	-0.1%	+0.9%
4	utl_rate = 0.1 (10%)	58%	-0.5%	-5.4%	-1.0%	+1.0%	-0.1%
5	t_layers = all	89%	-0.8%	-3.7%	-3.0%	-4.4%	+0.8%
6	t_mlp = Gate	81%	+1.3%	-0.4%	-1.0%	-1.8%	+1.1%
7	t_mlp = Up	49%	+1.0%	-5.8%	-1.7%	+0.8%	-0.8%
8	t_mlp = Down	17%	-9.3%	-16.3%	-13.1%	-12.0%	-6.0%
9	t_mlp = Gate and Down	30%	-1.7%	-0.3%	-0.6%	+1.5%	-0.6%
10	t_mlp = Down and Up	68%	-10.1%	-16.2%	-11.1%	-12.8%	-6.6%
11	t_mlp = Gate, Up, and Down	96%	-9.6%	-17.3%	-13.6%	-12.4%	-9.1%
12	t_attn = True	97%	-7.0%	-16.1%	-12.3%	-14.7%	-10.3%
13	n_out_pr = 10	87%	+0.6%	-2.7%	-1.9%	-0.5%	0.0%
14	n_out_pr = 25	88%	+0.6%	-2.7%	-1.9%	-0.5%	0.0%
15	n_out_pr = 100	89%	+0.6%	-2.7%	-1.9%	-0.5%	0.0%
16	n_out_pr = 500	91%	+0.6%	-2.7%	-1.9%	-0.5%	0.0%
17	n_iter = 10	93%	-1.55%	-2.9%	-2.95%	-5.2%	-1.25%
18	pr_rate = 0.005	84%	-0.8%	-2.7%	-0.7%	0.0%	-0.6%
19	pr_rate = 0.015	89%	-2.6%	-2.1%	-3.7%	-3.9%	-0.1%
20	n_iter = 10, pr_rate = 0.005	88%	-0.5%	-2.95%	-1.65%	-1.55%	-0.7%
21	n_iter = 3, pr_rate = 0.015	84%	-1.5%	-4.5%	-2.33%	-1.67%	+1.17%
22	n_iter = 1, pr_rate = 0.05	83%	-5.5%	-2.0%	-8.0%	-2.5%	-3.5%
23	Using none-twin prompts	82%	+0.1%	-10.0%	-8.4%	-6.8%	-4.4%
24	agr = False	60%	-1.0%	-4.5%	-1.0%	+1.5%	+2.0%
25	batch = single batch, b_size = 5	79%	-0.9%	-6.0%	-5.1%	-4.7%	-2.1%
26	batch = single batch, b_size = 25	88%	+1.2%	-2.6%	-1.9%	-3.3%	-0.4%
27	batch = multi-batch, b_size = 5	85%	-1.4%	-2.7%	-4.4%	-5.4%	-1.0%
28	batch = multi-batch, b_size = 25	88%	-0.6%	-3.6%	-2.3%	-3.3%	-0.9%
29	mean = top 3 tokens	98%	-7.2%	-20.1%	-17.7%	-15.1%	-19.2%
30	mean = top 6 tokens	93%	-0.9%	-2.4%	-1.7%	-1.5%	-0.3%
31	n_out_gen = 2, mean = top token	99%	-3.0%	-2.3%	-2.3%	-5.2%	-4.7%
32	n_out_gen = 2, mean = top 3 tokens	99%	-8.4%	-17.4%	-16.1%	-13.0%	-21.4%
33	n_out_gen = 2, mean = top 5 token	87%	-0.2%	-2.4%	-3.9%	-2.4%	-0.9%
34	n_out_gen = 2, mean = all (first out + last six input)	85%	+0.6%	+0.8%	-2.6%	-0.8%	-1.1%
35	n_out_gen = 6, mean = top token	80%	-1.8%	-6.5%	-5.2%	-6.2%	-3.3%
36	n_out_gen = 6, mean = top 3 tokens	92%	-5.6%	-12.2%	-4.1%	-7.6%	-3.4%
37	n_out_gen = 6, mean = top 5 token	93%	-0.7%	-11.7%	-3.7%	-7.4%	-2.2%
38	n_out_gen = 6, mean = all (five out + last six input)	90%	-2.7%	-2.5%	-4.0%	-4.7%	-3.1%
39	t_inp = all, mean = top token	84%	+0.5%	-3.7%	-2.5%	-2.8%	-2.1%
40	t_inp = all, mean = top 3 tokens	91%	+0.5%	-3.9%	-1.0%	-1.7%	-0.9%
41	t_inp = all, mean = top 5 tokens	91%	-0.1%	-4.5%	-1.7%	-2.4%	-0.7%
42	t_inp = all, mean = all (all input tokens)	85%	-1.6%	-3.5%	-1.5%	-4.8%	-4.8%
43	t_inp = all, n_out_gen = 2, mean = top token	97%	-3.7%	-4.0%	-0.9%	-4.5%	-4.9%
44	t_inp = all, n_out_gen = 2, mean = top 3 tokens	89%	-1.0%	-5.7%	-4.2%	-2.1%	-0.2%
45	t_inp = all, n_out_gen = 2, mean = top 5 tokens	87%	-0.9%	-2.5%	-3.5%	-2.0%	-1.3%
46	t_inp = all, n_out_gen = 2, mean = all	86%	-1.4%	-2.3%	-3.8%	-5.8%	-4.2%
47	t_inp = all, n_out_gen = 6, mean = top token	73%	-3.5%	-3.8%	-4.9%	-6.3%	-4.2%
48	t_inp = all, n_out_gen = 6, mean = top token	90%	-3.4%	-7.3%	-4.6%	-6.0%	-2.6%
49	t_inp = all, n_out_gen = 6, mean = top token	93%	+0.3%	-11.6%	-4.2%	-6.0%	-2.7%
50	t_inp = all, n_out_gen = 6, mean = top token	88%	-1.9%	-3.5%	-2.7%	-5.0%	-2.9%
51	t_inp = last	83%	-2.9%	-2.4%	-2.4%	-2.0%	+1.9%
52	t_inp = last, n_out_gen = 2, mean = top token	98%	-4.2%	-4.0%	-3.2%	-5.7%	-4.5%
53	t_inp = last, n_out_gen = 2, mean = all (top 2 tokens)	27%	-1.0%	+0.8%	-3.2%	-3.0%	-1.2%
54	t_inp = last, n_out_gen = 6, mean = top token	83%	-1.2%	-4.2%	-3.8%	-5.8%	-3.5%
55	t_inp = last, n_out_gen = 6, mean = top 3 tokens	90%	-5.4%	-10.4%	-4.5%	-7.3%	-2.8%
56	t_inp = last, n_out_gen = 6, mean = top 5 tokens	91%	-1.0%	-4.6%	-3.4%	-5.7%	-2.8%
57	t_inp = last, n_out_gen = 6, mean = all	89%	-3.2%	-11.0%	-4.3%	-6.8%	-3.7%
58	t_inp = None, n_out_gen = 2	97%	-3.6%	-3.8%	-3.9%	-5.0%	-5.3%
59	t_inp = None, n_out_gen = 6, mean = top token	79%	-1.2%	-3.7%	-4.0%	-5.7%	-3.3%
60	t_inp = None, n_out_gen = 6, mean = top 3 tokens	84%	-3.6%	-7.6%	-4.2%	-6.5%	-3.9%
61	t_inp = None, n_out_gen = 6, mean = all	87%	-1.6%	-0.4%	-2.3%	-5.8%	-3.8%

the safety mechanism of these models and also the ability of TwinBreak to a highly accurate and fine-grained safety parameter identification and pruning.

## 9.6 Incoherent Jailbreak Examples

Here we provide incoherence examples generated by poor configurations of TwinBreak and also Wei *et al.*’s set difference method.

Fig. 12 shows how choosing the wrong set of token activations for identification of safety parameters can result in incoherence and unexpected LLM outputs for TwinBreak.

Fig. 11 shows how the pruned model from Wei *et al.*’s [67] approach results in high utility degradation of the model hence resulting in low scores on StrongREJECT [57].

## 9.7 Additional Results on Comparison with Related Work

Here, we provide additional results for comparison between TwinBreak, directional ablation by Ardit *et al.* [4], and set difference by Wei *et al.* [67] in Tab. 21, Tab. 22, and Tab. 23, respectively. To compare with Ardit *et al.*, we expand the method to include new models, namely Gemma 2 and Qwen 2.5. We compare this method with TwinBreak on all the models and datasets that we used to in Sect. 4. Wei *et al.*’s approach only targets various LLaMA 2 models. In Tab. 23, we provide results with different configurations of the approach.

TwinBreak consistently outperforms Ardit *et al.*’s approach on LLaMA 2 and LLaMA 3.1 over all the datasets. On Qwen 2.5, TwinBreak achieves higher or similar ASRs on HarmBench, JailbreakBench, and AdvBench while Ardit *et al.*’s method slightly performs better on StrongREJECT. The same applies to Gemma 2. However, the difference is minimal.

Wei *et al.*’s set difference method is significantly weaker than TwinBreak. In addition, set difference requires several hours to compute importance scores while TwinBreak takes only several minutes to find safety parameters.

## 9.8 Accuracy on Utility Benchmarks for Different Dataset Sizes

The plots in Fig. 13 demonstrate the average accuracy of the utility benchmarks for the models at each dataset size. We can see that generally, the average accuracy is stable across various dataset sizes. However, we can observe some noticeable drops for specific dataset sizes. Specifically, we can see that for the dataset sizes of 60 and 70, LLaMA 2 shows a noticeable drop in average accuracies over all the benchmarks. This is due to the steam of nonsensical outputs as

discussed in Sect. 9.1. We could easily address this by increasing the utility parameter rate. Additionally, LLaMA 3.1 accuracy shows a significant drop for the dataset size of 60 on all the benchmarks. However, both models show stable results on the benchmarks for other dataset sizes. In addition, all the models show high average accuracies for dataset sizes of 80 and greater.

## 9.9 Different Models and Model Sizes

Tab. 24 and Tab. 25 show results on varied model sizes and families for TwinBreak and Directional Ablation [4] on StrongREJECT [57].