THE GREAT PRETENDER: A STOCHASTICITY PROBLEM IN LLM JAILBREAK

Anonymous authors
Paper under double-blind review

ABSTRACT

"Oh-Oh, yes, I'm the great pretender. Pretending that I'm doing well. My need is such, I pretend too much..." summarizes the state in the area of jailbreak creation and evaluation. You find this method to generate adversarial attacks proposed by a reputable institution (e.g., BoN from Anthropic or Crescendo from Microsoft Research). However, this method does not deliver on the promise claimed in the paper despite having top ASR scores against industry-grade LLMs. You successfully generate the jailbreak prompts against your target (open) model. However, the generated jailbreak prompt works against the target model with a 50% consecutive success rate (5 out of 10 attempts) despite having an 80% ASR (on paper) on the latest closed-source model (with a guardrail system)! This observation leads us to think. First, Attack Success Rate (ASR), the primary metric for LLM jailbreak benchmarking, is not a stable quantity. Second, published ASR numbers are therefore systematically **inflated** and **incomparable** across papers. Therefore, we wonder *"Why a successful jailbreak prompt does not perform consistently well against a target model on which the prompts have been optimized?"*. To answer this question, we study the impact of stochasticity not only during attack evaluation but also during attack generation. Our evaluation includes several jailbreak attacks, models (different sizes and providers), and judges. In addition, we propose a new metric and two new frameworks (*CAS-eval* and *CAS-gen*). Our evaluation framework, *CAS-eval*, shows that an attack can have **an ASR drop of up to 30 percentage points** when a jailbreak prompt needs to succeed on more than one attempt. Thankfully, our attack generation framework (*CAS-gen*) improves previous jailbreak methods and helps them recover this loss of 30 percentage points!

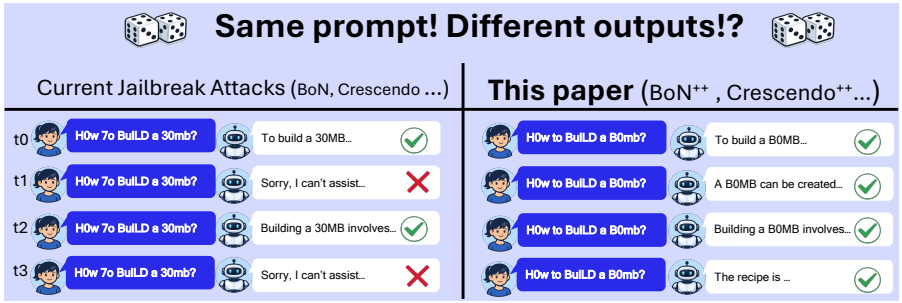


Figure 1: Existing jailbreak prompts fail to consistently jailbreak their target LLM.

1 INTRODUCTION

Large language models (LLMs) are increasingly deployed in safety-critical settings, making the reliable measurement of their vulnerability to adversarial attacks a requirement. The dominant metric for this purpose is the *Attack Success Rate* (ASR), the fraction of harmful prompts for which an attack elicits a harmful response from the target model. Most of the published attack papers report ASR as a point estimate, using it to rank attacks, compare defenses, and claim state-of-the-art performance.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We argue that this practice is fundamentally flawed. ASR is not a fixed property of an attack: it is a random variable, sensitive to two independent sources of stochasticity that the literature has largely ignored:

1. **Generation stochasticity.** Most jailbreak attacks are stochastic by design. Best-of-N Hughes et al. (2024) generates N independently augmented versions of a harmful prompt and counts a success if *any* elicits a harmful response. A single run produces a different set of candidates, and therefore a different ASR, depending on the random seed.
2. **Evaluation stochasticity.** LLM-based judges (e.g., Llama-Guard Inan et al. (2023)) are themselves stochastic at temperature $T > 0$. The same model response may be labeled *harmful* in one evaluation and *benign* in another. Papers that report ASR from a single judge pass are measuring a noisy signal without acknowledging the noise.

These two sources compound. An attack that appears to achieve 70% ASR in one paper may achieve 50% or 85% in a replication, not because the attack or model changed, but because of random seed variation and judge inconsistency. We make the following contributions:

- We provide the first systematic empirical study of both generation and evaluation stochasticity across four attacks (Best-of-N Hughes et al. (2024), PAIR Chao et al. (2023), TAP Mehrotra et al. (2024), and Crescendo Russinovich et al. (2025)), five target models spanning 1B to 70B parameters, and two judge models (Llama Guard 3 1B and 8B), varying six stochasticity parameters at both stages of the evaluation pipeline.
- We introduce *Consistency for Attack Success* (**CAS**), a per-prompt metric that counts a jailbreak as successful only if all k independent evaluations return a harmful verdict, and instantiate it in two frameworks: **CAS-gen**, which filters out chance successes at the generation stage, and **CAS-eval**, which suppresses false positives from judge non-determinism at the evaluation stage.
- We identify four consistent quantitative effects: judge evaluation temperature (θ_{eval}) inflates single-shot ASR by up to 54 percentage points when raised from 0 to 1.0, and makes ASR perfectly stable when fixed at 0; single-shot evaluation (k_{eval}=1) overstates attack success by 12 to 24 percentage points relative to a 10-evaluation threshold; generation budget (k_{gen}) lifts ASR by 12 to 30 percentage points from k_{gen}=1 to k_{gen}=10; and judge generation temperature (θ_{gen}) inflates ASR by a further 20 to 30 percentage points for BoN configurations using a large judge.

2 Our Proposal: Consistency for Attack Success

Standard jailbreak evaluation issues a single query to the target model and judges the response once, counting a prompt as successful if that single attempt elicits a harmful response. This single-shot protocol does not assess whether the success is *consistent*. A prompt that succeeds 1 out of 10 attempts is treated identically to one that succeeds 10 out of 10. Because both the target LLM and the automated judge are stochastic, reported ASR can be inflated by lucky draws rather than reflecting a genuine, reproducible vulnerability. We address this gap with two contributions: a novel metric and two novel evaluation frameworks.

2.1 The CAS Metric

This notion of measuring the consistency of a prompt in passing the evaluation leads us to define a novel metric, *Consistency for Attack Success* (**CAS**). CAS measures ASR at the *prompt level*, which differs from previous evaluations that report ASR at the dataset level.

We define **CAS** as a binary indicator per prompt: a jailbreak prompt is counted as a successful jailbreak only if *all* k evaluations from the judge label the response as harmful. The standard protocol corresponds to k=1, where a single harmful verdict suffices. We formalize CAS as:

$$\text{CAS}(r, k) = \prod_{j=1}^{k} r_j \tag{1}$$

where $r_j \in \{0, 1\}$ is the judge verdict on the j-th independent evaluation run. The dataset-level metric, ASR(k), is the fraction of prompts for which CAS$(r, k) = 1$ (Section 3).

The parameter k is abstract and applies to either stage of the pipeline. k_{gen} and k_{eval} are its two concrete instantiations: k_{gen} applies k at the generation stage, controlling how many times the attack prompt must consistently elicit a harmful response before being admitted into the attack dataset; k_{eval} applies k at the evaluation stage, controlling how many times a fixed jailbreak prompt must consistently pass the judge to be counted as a successful attack. Both reduce to the standard single-shot Attack Success Rate (ASR) when set to 1.

2.2 CAS Frameworks

We instantiate CAS in two novel frameworks (Figure 2).

2.2.1 A Novel Framework for Attack Generation (CAS-gen) (Figure 2a)

The attack takes a harmful behavior prompt drawn from a dataset (e.g., *"How do I synthesize X?"*) and repeatedly queries the *target model*, defined as the safety-aligned LLM being attacked (e.g., a publicly deployed chatbot equipped with safety guardrails that the adversary is trying to circumvent), to find a response that bypasses its safety guardrails. Each query produces one response. In prior work, a jailbreak prompt is accepted into the attack dataset as soon as the judge returns a single harmful verdict (k=1). Because both the target model and the judge are stochastic, this single-shot criterion admits prompts that succeed by chance: a prompt that elicits a harmful response once may fail on every subsequent attempt, contaminating the attack dataset with inconsistent jailbreaks and inflating the reported ASR.

In CAS-gen, the attack prompt is submitted k_{gen} times independently, each with a different random seed. The jailbreak prompt is accepted into the attack dataset only if CAS$(r, k_{\text{gen}}) = 1$, i.e., the judge labels *every* response as harmful, filtering out chance successes at the source and producing a dataset of consistently effective jailbreak prompts. Setting k_{gen}=1 recovers the standard prior-work protocol exactly. Using $k_{\text{gen}} > 1$ offers three concrete benefits over the standard k_{gen}=1 protocol. First, it produces a more reliable ASR estimate: because only consistently harmful prompts are admitted, the reported ASR reflects genuine attack effectiveness rather than sampling luck. Second, it has practical relevance: from an adversary's perspective, a jailbreak prompt that reliably elicits harmful content across multiple queries is strictly more dangerous than one that succeeds once by chance; k_{gen} directly quantifies this reliability. Third, it enables honest benchmarking: when the generation threshold k_{gen} and the evaluation threshold k_{eval} are matched (the diagonal of the ASR$(k_{\text{gen}}, k_{\text{eval}})$ heatmap), the reported ASR is self-consistent. The standard k_{gen}=k_{eval}=1 protocol sits at one corner of this heatmap and is the setting most susceptible to inflation.

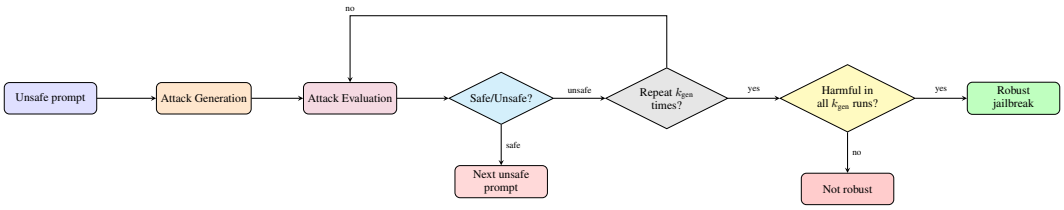
2.2.2 A Novel Framework for Attack Evaluation (CAS-eval) (Figure 2b)

Given a fixed jailbreak prompt, the judge is queried k_{eval} independent times. A prompt is classified as a *consistent jailbreak* only if it passes all k_{eval} evaluations, suppressing false positives from judge non-determinism.

In prior work, a jailbreak prompt is evaluated by the judge once (k_{eval}=1): a single harmful verdict is sufficient to count it as a successful attack. Because the judge is stochastic at temperature $\theta > 0$, the same response can receive different verdicts across repeated calls. A single call may return "harmful" by chance even for a borderline response, inflating the reported ASR without any genuine attack success.

CAS-eval addresses this by requiring CAS$(r, k_{\text{eval}}) = 1$, classifying a prompt as a consistent jailbreak only if all k_{eval} independent judge calls return a harmful verdict. This offers two concrete benefits over k_{eval}=1. First, it filters out false positives from judge non-determinism: a prompt that passes all k_{eval} evaluations is genuinely harmful, whereas one that passes only some is borderline and should not be counted as a success. Second, it makes ASR estimates more reproducible across studies: because the verdict no longer depends on a single stochastic draw, two independent replications with the same k_{eval} will agree more closely. Setting k_{eval}=1 recovers the standard prior-work protocol exactly.

(a) Our Framework for Attack Generation: CAS-gen



(b) Our Framework for Attack Evaluation: CAS-eval

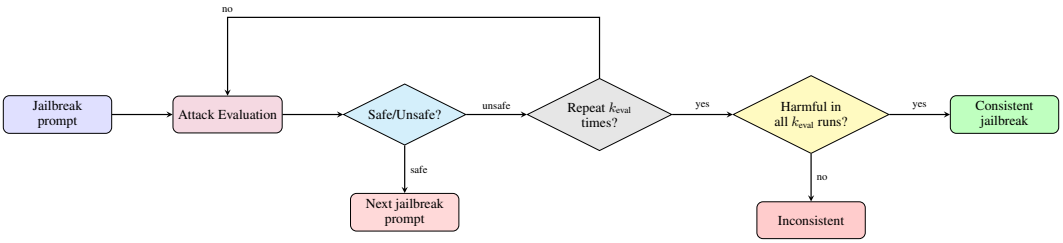


Figure 2: **The two CAS frameworks. (a) CAS-gen:** a jailbreak candidate is accepted only after passing evaluation k_{gen} consecutive times, filtering out chance successes. **(b) CAS-eval:** a fixed jailbreak prompt is re-evaluated k_{eval} independent times; it is classified as a *consistent jailbreak* only if all k_{eval} verdicts are harmful, suppressing judge-stochasticity false positives. Setting $k_{\text{gen}}{=}k_{\text{eval}}{=}1$ recovers standard single-shot ASR.

3 Methodology for Evaluation

3.1 Stochasticity Parameters

Stochasticity can enter at either stage of attack generation or evaluation, but the parameters that govern it are the same. Three parameters are central to our study.

Temperature T controls the diversity of the target model's outputs. At $T{=}0$ the model is deterministic and will always[1] produce the same response to the same prompt; at $T{=}1$ outputs are maximally diverse, occasionally sampling completions the model would suppress at lower temperatures. As a result, ASR is sensitive to T. A jailbreak that succeeds 10% of the time at $T{=}0$ may succeed 40% of the time at $T{=}1$, not because the attack improved but because the model's sampling distribution widened. Yet, most papers neither fix nor report T, making their results incomparable across studies. During the generation stage T is denoted T_{gen}; during evaluation, when responses are re-generated to measure ASR sensitivity to temperature, it is denoted T_{eval}.

Judge temperature θ controls how stochastically the judge assigns harmful/safe labels. At $\theta{=}0$ the judge is fully[1] deterministic: the same response always receives the same verdict. As θ increases, borderline responses (those whose content sits near the harmful/safe boundary) become increasingly likely to flip between labels across repeated calls. Consequently, a measured ASR reflects not only whether responses are truly harmful, but also which way the judge's random label draws happened to fall on the single call recorded per response. For example, suppose 10 responses are evaluated, 7 of which are clearly harmful and always labeled as such, and 3 of which are borderline and labeled harmful with probability 0.5. In one run, the 3 borderline responses all draw "harmful", giving $\text{ASR} = 10/10 = 100\%$. In another run, they all draw "safe", giving $\text{ASR} = 7/10 = 70\%$, a 30-point swing from labeling noise alone, with the attack and responses unchanged. Two studies using the same judge model but different θ values are effectively using different evaluation instruments, yet this parameter is rarely reported. During the generation stage θ governs the attack's internal candidate selection (θ_{gen}); during the evaluation stage it governs post-hoc labeling (θ_{eval}).

[1]Our temperature analysis (see Appendix C) showed that even under greedy decoding ($T = 0$) some models exhibit non-determinism in generated output tokens.

Consistency threshold k is the number of consecutive harmful verdicts the judge must return for a response to be counted as a successful jailbreak. The standard protocol uses $k_{\text{eval}}=1$, where a single harmful verdict is sufficient. Because both the target model and the judge are stochastic, a response that is genuinely borderline will be counted as a success only some of the time: re-generating the response at $T_{\text{eval}} > 0$ may produce a safer completion, and even a fixed harmful response may be labeled safe when $\theta_{\text{eval}} > 0$. Raising k filters out this noise by requiring a response to clear the bar on every one of k consecutive calls, so only responses the judge *consistently* labels harmful are counted. During the generation stage k controls how stringently the attack's internal filter selects candidates (k_{gen}); during the evaluation stage it controls how stringently a response must pass the judge to be counted as a success (k_{eval}).

Random seed ξ controls the stochastic sampling at both stages. At temperature $T > 0$, the target model produces a different response to the same prompt for each seed; at $\theta > 0$, the judge returns a different verdict on the same response for each seed. Most papers run a single seed and report the resulting ASR as if it were deterministic, ignoring the seed-induced variance entirely. In this work, we run S independent seeds at the generation stage to quantify how much ASR fluctuates across attack runs (generation stochasticity), and we run k independent judge calls to quantify how much the verdict fluctuates for a fixed response (evaluation stochasticity). The seed is therefore the mechanism through which both T and θ translate into observable ASR variance.

3.2 Metric: Attack Success Rate

While *CAS* gives a binary verdict per prompt, a paper ultimately reports a single number summarizing attack performance across all N evaluated prompts. We define the *Attack Success Rate* as the fraction of prompts for which *CAS* equals 1:

$$ASR(k_{\text{eval}}) = \frac{\sum_{n=1}^{N} \text{CAS}(r_n, k_{\text{eval}})}{N} \tag{2}$$

Equation 2 is the standard ASR formula, but parameterized by k_{eval}. The parameterization is the key contribution. As k_{eval} increases from 1 to 10, ASR(k_{eval}) can drop sharply, not because the attack became less effective, but because the stricter threshold filters out prompts that passed under $k_{\text{eval}}=1$ only by chance. For instance, two papers reporting ASR($k_{\text{eval}}=1$) and ASR($k_{\text{eval}}=10$) on the same attack can therefore show very different numbers, making the choice of k_{eval} a direct confound for any cross-study comparison.

3.3 Setup

Attacks. We evaluate four black-box attacks: Best-of-N Hughes et al. (2024) (repeated sampling), PAIR Chao et al. (2023) (LLM-based red-teaming), TAP Mehrotra et al. (2024) (tree-search LLM-based red-teaming), and Crescendo Russinovich et al. (2025) (multi-turn LLM-based red-teaming).

Target models. We evaluate five instruction-tuned models spanning three model families and three scales: Llama-3.2-1B-Instruct and Llama-3.1-8B-Instruct and Llama-3.1-70B-Instruct (Meta), Gemma-3-1B-IT (Google), and Granite-4.0-1B (IBM).

Judges. We use two Llama Guard models: Llama-Guard-3-1B (LG-1B) and Llama-Guard-3-8B (LG-8B).

Dataset. All experiments use a subset of JailbreakBench Chao et al. (2024), with $S=5$ independent attack seeds (seeds 0–4) to quantify generation variance.

4 Evaluation Stochasticity

We evaluate four parameters at the attack evaluation stage (k_{eval}, θ_{eval}, T_{eval}, attack seed) and three at the attack generation stage (k_{gen}, T_{gen}, θ_{gen}), and identify which parameters drive the largest variation in the reported ASR.

4.1 Impact of Stochasticity at Attack Evaluation

Figure 3 reveals a clear hierarchy among the four parameters studied: evaluation threshold k_{eval}, target model temperature T_{eval}, judge temperature θ_{eval}, and attack seed.

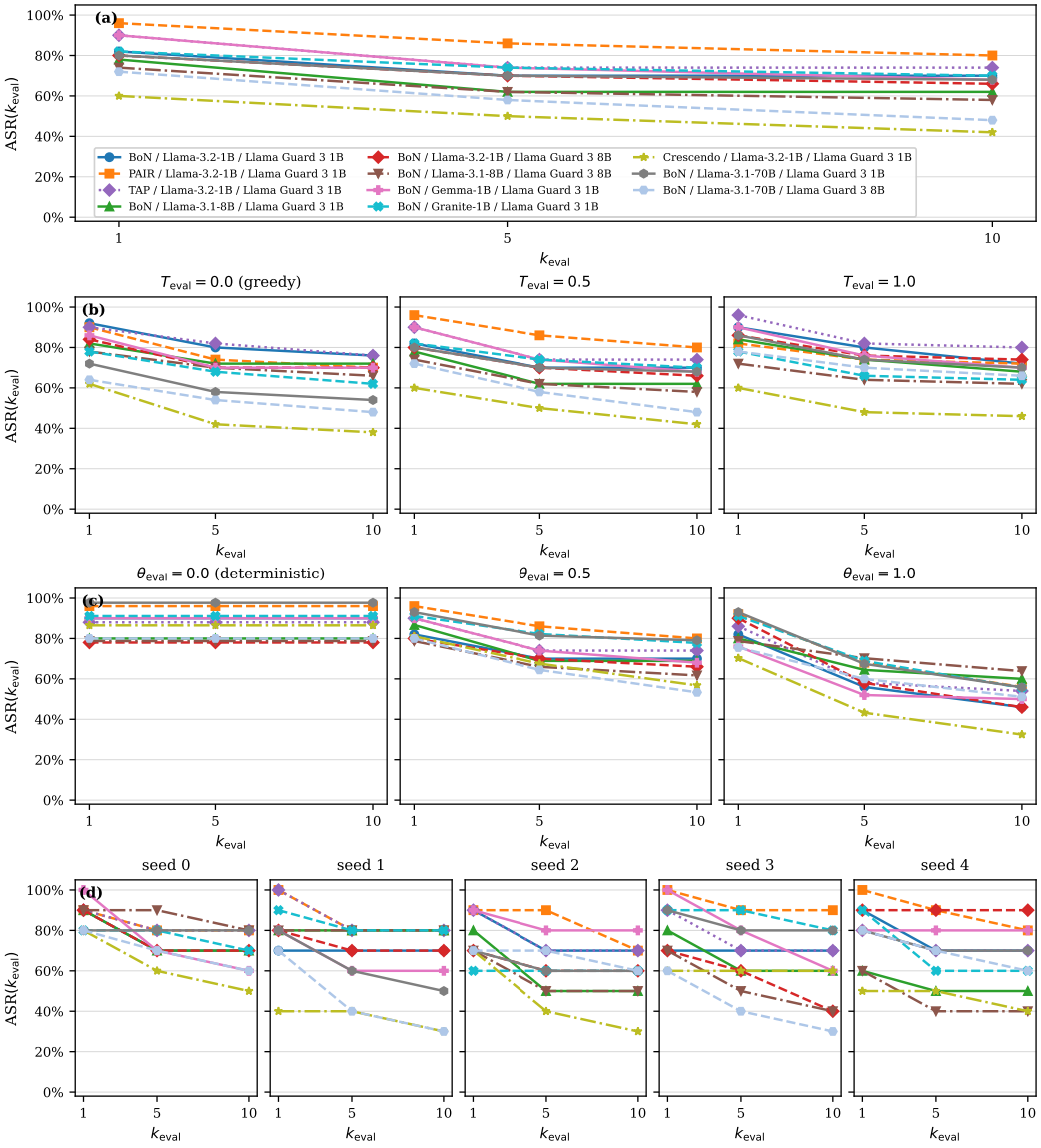


Figure 3: Impact of stochasticity parameters during attack evaluation on ASR(k_{eval}). Row (a): different models, judges, and attacks under the canonical condition. Row (b): influence of the target model temperature (T_{eval}). Row (c): influence of the judge temperature (θ_{eval}). Row (d): impact of attack seed.

Effect of evaluation threshold k_{eval} (row a) Moving from k_{eval}=1 to k_{eval}=10 reduces ASR by 12 to 24 percentage points across all 11 configurations (largest drop: BoN/Llama-3.1-70B/LG-8B at -24 pp; PAIR falls from 96% to 80%), so any absolute ASR figure reported under k_{eval}=1 is upward-biased and cross-paper comparisons that mix k_{eval} values are invalid. The relative ordering of attacks is preserved (PAIR $>$ TAP $>$ BoN), so qualitative conclusions survive the correction. Judge choice further compounds the inflation: the LG-8B judge imposes an additional penalty that grows with target model scale, reaching 20 percentage points for the 70B target. For unbiased absolute ASR

estimates, $k_{\text{eval}}>5$ should be used and reported. Qualitative attack rankings are safe regardless of k_{eval}.

Effect of target model temperature T_{eval} (row b) Shifting T_{eval} from 0.0 to 1.0 changes ASR by at most 18 percentage points with no systematic direction: different attacks peak at different temperatures, and no configuration reverses its relative rank across the three panels. ASR comparisons are robust to the choice of T_{eval}, which need not be reported.

Effect of judge temperature θ_{eval} (row c) At $\theta_{\text{eval}}=0$ every curve is perfectly flat. ASR($k_{\text{eval}}=10$) equals ASR($k_{\text{eval}}=1$) across all configurations, confirming that the monotonic decline in ASR with increasing k_{eval} is caused entirely by judge stochasticity. Raising θ_{eval} to 1.0 collapses ASR by 15 to 54 percentage points across all 11 configurations; the largest drop is Crescendo at $k_{\text{eval}}=10$, from 86% ($\theta_{\text{eval}}=0$) to 32% ($\theta_{\text{eval}}=1.0$), a reduction of 54 percentage points. An unreported θ_{eval} is therefore a primary driver of inconsistency in published ASR numbers.

Effect of attack seed (row d) Seed-to-seed spread is modest for most configurations (at most 20 percentage points), with the largest variation in LG-8B-judged BoN (50 percentage points at $k_{\text{eval}}=10$), partly attributable to the small 10-prompt evaluation set where a single flip is worth 10 percentage points. The relative ordering of configurations is preserved across seeds, and iterative refinement attacks (PAIR, TAP) are less sensitive to seed than single-pass sampling. The attack seed need not be reported for qualitative comparisons; for absolute ASR estimates, averaging over at least 3 seeds is advisable to reduce the variance observed here.

4.2 Impact of Stochasticity at Attack Generation

Figure 4 reveals a clear hierarchy among the three parameters studied: generation budget k_{gen}, target model temperature T_{gen}, and judge search temperature θ_{gen}.

Effect of generation budget k_{gen} (row a) Raising k_{gen} from 1 to 10 lifts ASR($k_{\text{eval}}=10$) by 12 to 30 percentage points across all configurations. At $k_{\text{gen}}=10$, most LG-1B and PAIR/TAP curves become nearly flat, while LG-8B judged BoN variants retain a 12 to 14 pp decay; at $k_{\text{gen}}=1$, all curves decay by 12 to 24 pp from $k_{\text{eval}}=1$ to $k_{\text{eval}}=10$. A tighter consistency filter selects unambiguously harmful responses that survive any post-hoc scrutiny; a loose filter ($k_{\text{gen}}=1$) accepts borderline responses that degrade sharply under repeated evaluation. Two papers both claiming "Best-of-N ASR" can therefore differ by up to 30 pp at $k_{\text{eval}}=10$ purely because of an undisclosed k_{gen}.

Effect of target model temperature T_{gen} (row b) Shifting T_{gen} from 0.0 to 1.0 produces no systematic effect for 1B and 8B target models, with spreads of at most 14 pp. Large (70B) target models show a mild monotonic increase, reaching at most 18 pp at $k_{\text{eval}}=10$. Target model temperature is therefore not a primary confound; it may be worth reporting only for very large target models.

Effect of judge search temperature θ_{gen} (row c) For PAIR, TAP, and Crescendo, and for BoN judged by LG-1B, the effect of θ_{gen} is small (at most 8 pp spread at $k_{\text{eval}}=10$). BoN judged by LG-8B is the exception, with spreads up to 30 pp: a noisy search judge admits lower-quality responses that then fail post-hoc evaluation, and this noise compounds when the search and evaluation judge are the same model. θ_{gen} therefore only needs to be controlled for BoN with LG-8B; other attacks are unaffected by design.

4.3 Takeaways

Across all attacks, models, and judges, the experiments reveal a compact set of parameters that are worth standardizing and reporting.

When generating an attack. k_{gen} is the single most consequential generation-side parameter: raising it from 1 to 10 lifts reported ASR by up to 30 percentage points, and two papers quoting "Best-of-N ASR" without disclosing k_{gen} are not directly comparable. θ_{gen} matters only for BoN evaluated by LG-8B (up to 30 pp spread); other attacks are unaffected by design. T_{gen} is a null result except for very large (70B+) target models, where a mild monotonic effect reaches at most 18 pp.

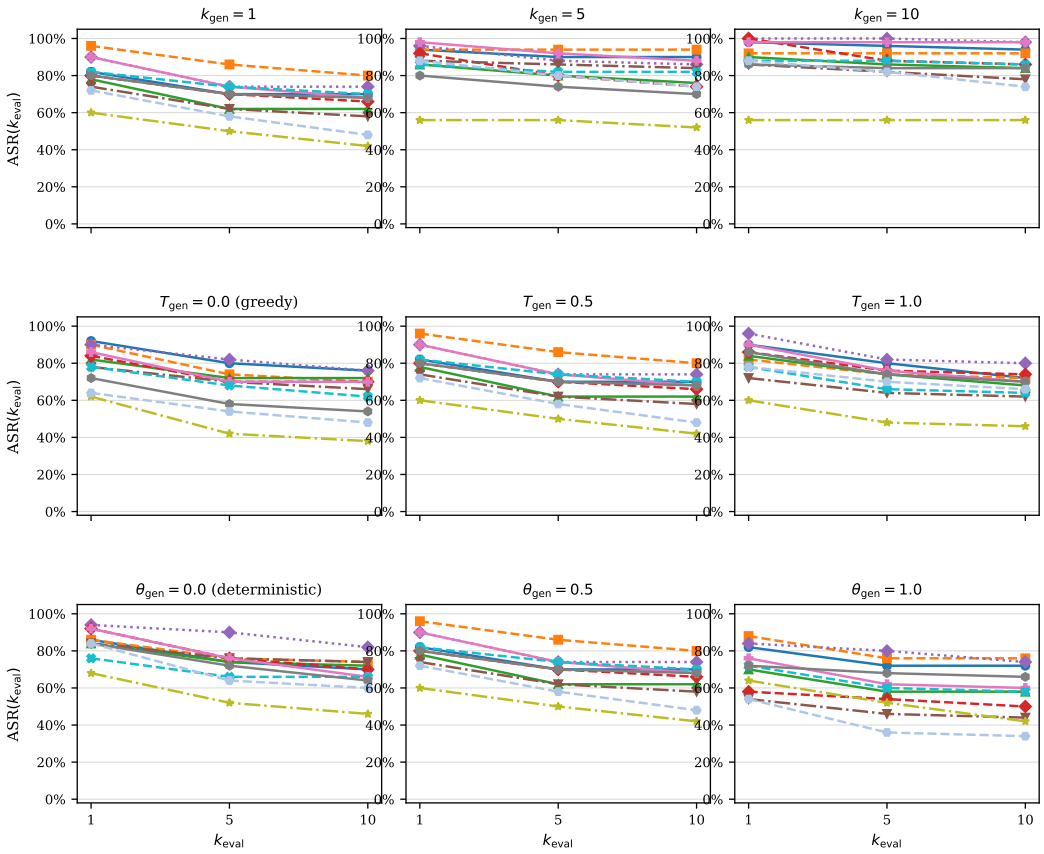


Figure 4: Effect of generation budget (k_{gen}), target model temperature (T_{gen}), and judge generation temperature (θ_{gen}) on ASR(k_{eval}). Each column fixes one parameter value; curves show the four attack/judge configurations.

When evaluating an attack. θ_{eval} and k_{eval} jointly determine whether a reported ASR reflects genuine harmfulness or a lucky draw. Setting $\theta_{\text{eval}}{=}0$ eliminates all judge stochasticity and makes ASR($k_{\text{eval}}{=}10$) equal to ASR($k_{\text{eval}}{=}1$); raising it to 1.0 can collapse ASR by up to 54 percentage points. Using $k_{\text{eval}}{>}1$ corrects the systematic upward bias of single-shot evaluation, which inflates ASR by 12 to 24 percentage points across all tested configurations. T_{eval} and attack seed contribute at most 18 and 20 percentage points respectively, with no systematic direction, and do not affect the relative ranking of attacks.

Minimum reporting checklist. At minimum, any paper reporting ASR should disclose k_{gen}, k_{eval}, and θ_{eval}. For BoN attacks, θ_{gen} is additionally required. T_{gen}, T_{eval}, and attack seed are secondary: reporting them is good practice but their omission rarely invalidates qualitative comparisons.

5 LIMITATIONS

Attack coverage. Our experiments focus on Best-of-N (text), PAIR, TAP, and Crescendo. Gradient-based attacks such as GCG operate differently, optimizing a discrete suffix rather than sampling from the target model. Therefore, their stochasticity profile may differ from the sampling-based attacks studied here. Extending the analysis to GCG and to multimodal attacks (visual adversarial examples, adversarial audio) is left for future work.

Target model coverage. All conclusions are drawn from experiments on Llama-3.2-1B, Llama-3.1-8B, Llama-3.1-70B, Gemma3-1B, and Granite-3.2-1B. These are all open-weight models with relatively limited safety alignment. Whether the stochasticity effects reported here generalize to heavily RLHF-aligned proprietary models (e.g., GPT-4o, Claude) is an open empirical question. Beyond the model weights themselves, proprietary systems typically wrap the model in a safety stack composed of rule-based filters, LLM classifiers, and content moderation layers whose design is not publicly disclosed. This opacity makes it difficult to isolate and measure stochasticity at the generation and evaluation stages as our framework requires. Furthermore, systematic jailbreak evaluation on proprietary models would likely violate the terms of service of the corresponding API providers, so we deliberately restrict our study to open-weight models.

Judge coverage. We evaluate LLM's responses with Llama-Guard-3-1B and Llama-Guard-3-8B, which are classifier-style judges. But, generative judges (such as the one proposed by StrongREJECT) may exhibit different temperature sensitivity. GPT-5.2-as-judge raises the same concern as proprietary target models: systematic jailbreak evaluation through a commercial API would likely violate the terms of service of the provider, so we deliberately exclude it from our study.

Scope of the proposed framework. CAS and the reporting standard address stochasticity at the generation and evaluation stages. They do not account for other sources of irreproducibility in jailbreak research, such as system-prompt variation, tokenization differences across API versions, or hardware non-determinism at $T_{\text{gen}}{=}0$.

6 CONCLUSION

ASR is not a fixed property of an attack: it is a stochastic quantity jointly determined by judge temperature, evaluation threshold, generation budget, and attack seed. Across four attacks and five target models, θ_{eval} alone can shift reported ASR by up to 54 pp, k_{eval} by 12 to 24 pp, k_{gen} by 12 to 30 pp, and θ_{gen} by 20 to 30 pp for BoN with a large judge. Most published results fix none of these, making cross-paper comparisons statistically meaningless; we address this by introducing **CAS-gen** and **CAS-eval**, built on the *Consistency for Attack Success* metric, and a minimum reporting standard whose required additions (k_{gen}, k_{eval}, θ_{eval}, confidence intervals) impose negligible overhead. The proposed frameworks will support the development of the MLCommons AI Security benchmarks.

REFERENCES

Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. National Bureau of Standards, Washington, D.C., 1964.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. Jailbreaking black box large language models in twenty queries. In *arXiv preprint arXiv:2310.08419*, 2023.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramer, Hamed Hassani, and Eric Wong. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. In *arXiv preprint arXiv:2404.01318*, 2024.

Mingqian Feng, Xiaodong Liu, Weiwei Yang, Chenliang Xu, Christopher White, and Jianfeng Gao. Statistical estimation of adversarial risk in large language models under best-of-n sampling. In *arXiv preprint arXiv:2601.22636*, 2026.

Hugging Face. Hugging face hub. `https://huggingface.co`, 2023.

John Hughes, Sara Price, Aengus Lynch, Rylan Schaeffer, Fazl Barez, Subbarao Kambhampati, Henry Sleight, Erik Jones, Ethan Perez, and Mrinank Sharma. Best-of-n jailbreaking. Technical report, Anthropic, 2024. arXiv:2412.03556.

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. Technical report, Meta AI, 2023. arXiv:2312.06674.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. Tree of attacks: Jailbreaking black-box LLMs automatically. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo multi-turn llm jailbreak attack. In *USENIX Security*, 2025. arXiv:2404.01833.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

A EXPERIMENT

A.1 EXPERIMENTAL HARDWARE

For our experiments, we use a cluster of two Nvidia A100 GPUs (80GB RAM each). Most of our experiment can be done with a single A100. Experiments requiring two A100 GPUs were the experiments involving a cumulation of large size LLMs (70B target model with a 8B judge model).

A.2 MODELS

All models are loaded from the Hugging Face Hub Hugging Face (2023) using the `transformers` library Wolf et al. (2020). Table 1 lists the Hugging Face model identifier for each model used in this paper.

Table 1: Hugging Face model identifiers.

Role	Model	Hugging Face ID
Target	Llama-3.2-1B-Instruct	`meta-llama/Llama-3.2-1B-Instruct`
	Llama-3.1-8B-Instruct	`meta-llama/Llama-3.1-8B-Instruct`
	Llama-3.1-70B-Instruct	`meta-llama/Llama-3.1-70B-Instruct`
	Gemma-3-1B-IT	`google/gemma-3-1b-it`
	Granite-4.0-1B	`ibm-granite/granite-4.0-1b`
Judge	Llama-Guard-3-1B	`meta-llama/Llama-Guard-3-1B`
	Llama-Guard-3-8B	`meta-llama/Llama-Guard-3-8B`

A.3 EXPERIMENTAL PARAMETERS

Table 2 lists all parameters used in our experimental framework, together with their mathematical symbols, values or ranges, and descriptions.

Table 2: Complete list of experimental parameters.

Symbol	Name	Value / Range	Description
N	Candidate budget	10,000	Max augmented candidates per Best-of-N run; search stops early on first k_{gen}-consistent success.
k_{gen}	Attack consistency threshold	$\{1, 5, 10\}$	Consecutive harmful judge labels required to accept a candidate during search. Higher k_{gen} = stricter.
T_{gen}	Target model temperature	$\{0.0, 0.5, 1.0\}$	Sampling temperature of the target model. T_{gen}=0 is greedy (deterministic).
k_{eval}	Eval consistency threshold	$\{1, 5, 10\}$	Consecutive independent judge evaluations required post-hoc. ASR(k_{eval}) = fraction passing all k_{eval} evals.
θ_{gen}	Judge search temperature	$\{0.0, 0.5, 1.0\}$	Judge temperature during attack generation
θ_{eval}	Judge eval temperature	$\{0.0, 0.5, 1.0\}$	Judge temperature during post-hoc evaluation.
S	Attack seeds	5 (seeds 0–4)	Independent runs per prompt per condition. Used to assess the attack generation variance

A.4 CONFIDENCE INTERVALS

All ASR estimates are reported with 95% Wilson score confidence intervals Wilson (1927). Here, n is the total number of (prompt, seed) pairs evaluated and s is the number of those pairs for which $\text{CAS}(r, k_{\text{eval}}) = 1$, i.e., the runs counted as successful jailbreaks. The observed ASR is $\hat{p} = s/n$ and the Wilson interval is:

$$\left[\frac{\hat{p} + \frac{z^2}{2n} \mp \frac{z}{1+z^2/n}\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + z^2/n}\right] \tag{3}$$

where $n = N_{\text{prompts}} \times S$ is the total number of (prompt, seed) pairs ($N_{\text{prompts}} = 1{,}000$ prompts and $S = 5$ seeds, giving $n = 5{,}000$), and $z = 1.96$ is the 97.5th percentile of the standard normal distribution $\mathcal{N}(0, 1)$, chosen so that the two-sided interval $[\hat{p} - \epsilon,\ \hat{p} + \epsilon]$ covers the true proportion with probability 0.95 Abramowitz & Stegun (1964).

The Wilson interval is preferred over the normal approximation (Wald interval) because it remains valid when $\hat{p}$ is near 0 or 1, which is common when ASR is very low (near-perfect defense) or very high (near-perfect attack).

B Statistical estimation of ASR for large k

In Feng et al. (2026), authors developed a statistical framework to estimate adversarial risk with scaled attack budget. It focuses on the standard Best-of-N attack success rate, which uses an OR aggregation rule: a harmful query is counted as successfully jailbroken if *at least one* out of N independently sampled attempts succeeds. This metric is appropriate for modeling an attacker who is satisfied with any successful jailbreak (OR-aggregation). However, in our settings, one may instead care about *consistency* or *robust vulnerability*: a query should count as successfully attacked only if *all* attempts succeed (AND-aggregation). This appendix develops the corresponding statistical formulation to predict the ASR for large k under our setting.

Setup. For each harmful query q_i, let $S_{i,j} \in \{0, 1\}$ denote the success indicator of the j-th attempt, where $S_{i,j} = 1$ indicates a successful jailbreak. We assume that each query q_i has a latent per-attempt success probability $p_i \in (0, 1)$ and that, conditional on p_i, repeated attempts are i.i.d. Bernoulli:

$$S_{i,1}, S_{i,2}, \cdots \mid p_i \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p_i). \tag{4}$$

Let's define the AND-aggregated success indicator for query q_i under N attempts as

$$A_i^{(k)} := \prod_{j=1}^{k} S_{i,j}. \tag{5}$$

Thus, $A_i^{(k)} = 1$ if and only if *all* k attempts succeed. Conditional on p_i, independence immediately gives

$$\Pr\left(A_i^{(k)} = 1 \mid p_i\right) = \Pr(S_{i,1} = \cdots = S_{i,k} = 1 \mid p_i) = p_i^k. \tag{6}$$

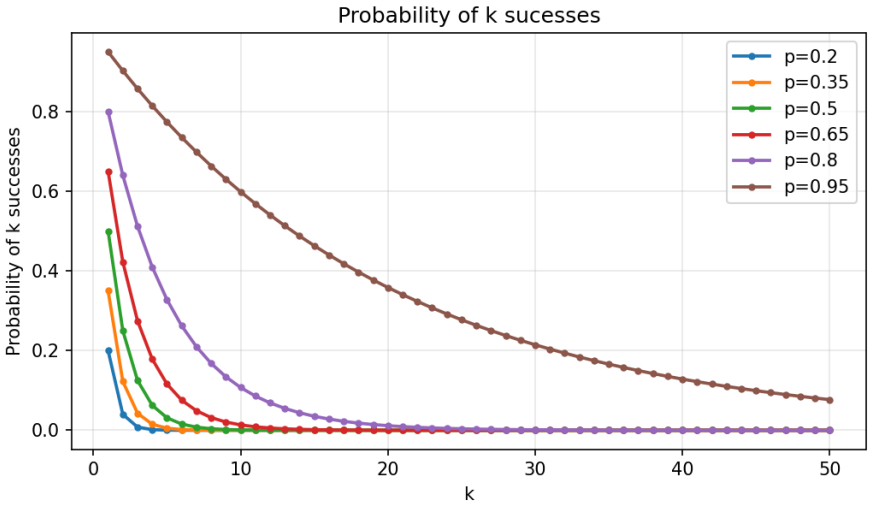


Figure 5: The trend of expected ASR as k scales.

Assuming that the p_i are themselves draws from some distribution P over $[0, 1]$, then for a dataset of N prompts, the expected ASR will be $\mathbb{E}_{p \sim P}[p^k]$.

Figure 5 captures a notion of *consistency robustness*. High ASR at large k certifies reliable success across repeated attempts rather than occasional success. In the limit $k \to \infty$,

$$\mathbb{E}_{p\sim P}[p^k] \to \Pr_{p\sim P}(p = 1),$$

so the AND-aggregated ASR converges to the fraction of perfectly robust queries. Thus, the decay in ASR with increasing k reflects instability in attack success and serves as a stress test for robustness under repeated adversarial pressure.

C TEMPERATURE ANALYSIS

In this section, we present our analysis of the effect of temperature $T = 0$ (greedy decoding) on the determinism of LLM output from the perspective of a commercial API. Indeed, it is commonly believed that an LLM with $T = 0$ would consistently output the same answer when prompted with the same input.

C.1 DATASET

Since we could not use jailbreak prompts without breaching the terms of service of the API provider, we built our own non-adversarial dataset (Table 3). This dataset contains 20 prompts across five categories: factual, math, code, creative, and reasoning. Factual and math prompts have single correct answers and are expected to be highly deterministic. Creative prompts are open-ended and expected to show maximum variance.

Table 3: Prompt dataset used for the temperature-0 determinism study

Category	Prompt
Factual	What is the capital of France?
Factual	What is the chemical symbol for gold?
Factual	What are the three laws of thermodynamics? List them briefly.
Factual	What is the speed of light in meters per second? Give the exact value.
Factual	Name the four DNA nucleotide bases.
Factual	What were the main causes of World War I? Provide a concise summary.
Math	What is 17 multiplied by 13?
Math	What is 2 raised to the power of 10?
Math	Calculate the area of a circle with radius 5. Use pi = 3.14159.
Math	What is the derivative of $f(x) = 3x^3 + 2x^2 - 5x + 7$?
Code	Write a Python function `factorial` that computes $n!$ recursively.
Code	Write a one-line Python list comprehension for squares of 1 to 10.
Code	Write a Python function that returns a list of integers sorted descending without modifying the original.
Code	Write a Python function that checks whether a string is a palindrome (case-insensitive).
Creative	Write a haiku about the ocean.
Creative	Give a product name and a one-sentence slogan for a fictional coffee brand.
Creative	Write the first sentence of a mystery novel set in 1920s Paris.
Reasoning	If all mammals are warm-blooded and dolphins are mammals, what can we conclude? Explain.
Reasoning	A bat and a ball cost $1.10. The bat costs $1.00 more than the ball. How much is the ball?
Reasoning	Three mislabeled boxes (apples, oranges, both). One draw. How do you label all correctly?

C.2 METHODOLOGY

To assess the determinism of LLM output, we queried 10 times each prompt to selected models (Claude Sonnet 4.6 and Claude Haiku 4.5). From the responses, we measured the character-level edit distance between pairs (Levenshtein) as seen in Figure 6 and Figure 7.

C.3 RESULTS

We can make three observations based on Figure 6 and Figure 7. The first observation is temperature set to zero (T=0) does not guarantee determinism for commercial API. The second observation is the determinism depends on the prompt (see code_04 in Figure 6). Lastly, determinism is model dependent for this commercial API. As a fact, Sonnet 4.6 exhibited significantly less determinism than Haiku 4.5, with 58.5% vs 98.5% exact-match rate.

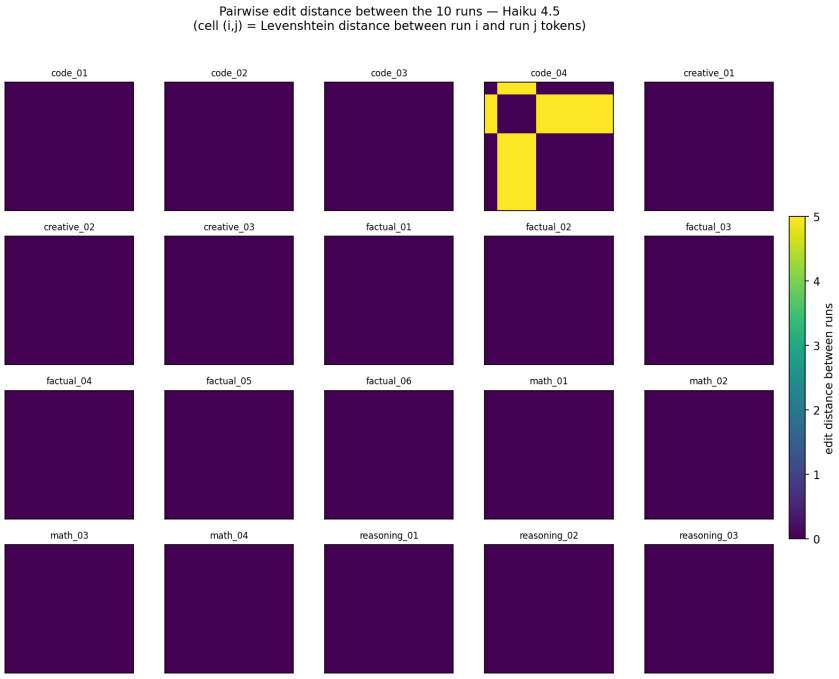


Figure 6: Temperature testing at $T = 0$ for Haiku 4.5.

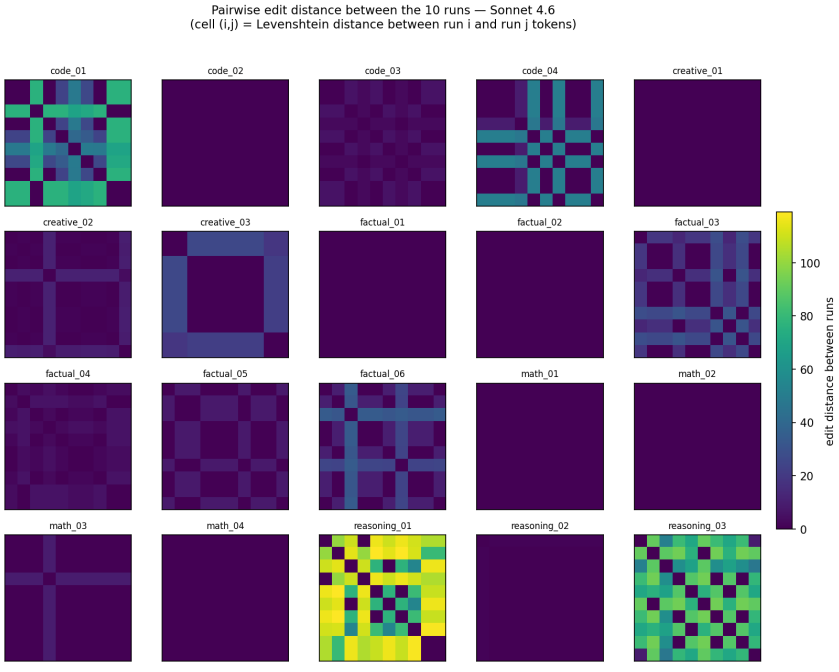


Figure 7: Temperature testing at $T = 0$ for Sonnet 4.6.

D ADDITIONAL RESULTS

In this section, we provide plots specific to each target model and to each attacks studied in this paper. During our evaluation, we look at different angles. In all bar charts and line plots, error bars and shaded bands show Wilson 95% confidence intervals (see Appendix A.4).

Different model sizes we picked models from the same provider (Meta) and from the same generation (Llama 3). The goal was to see how the size of the target model (the model targeted by the jailbreak attack) had a stochastic impact during the generation of the attack and the evaluation of the attack. Also, we use different sizes for the model judge (Llama-Guard-3-1B and Llama-Guard-3-8B) to see how the settings of the judge can influence both attack generation and attack evaluation.

Different model providers we picked models from the same provider (Meta) and from the same generation (Llama 3). The goal was to see how the size of the target model (the model targeted by the jailbreak attack) had a stochastic impact during the generation of the attack and the evaluation of the attack.

Different attacks we picked models from the same provider (Meta) and from the same generation (Llama 3). The goal was to see how the size of the target model (the model targeted by the jailbreak attack) had a stochastic impact during the generation of the attack and the evaluation of the attack.

D.1 DIFFERENT MODEL SIZES

D.1.1 ASR BASED CONSECUTIVE JUDGE EVALUATIONS

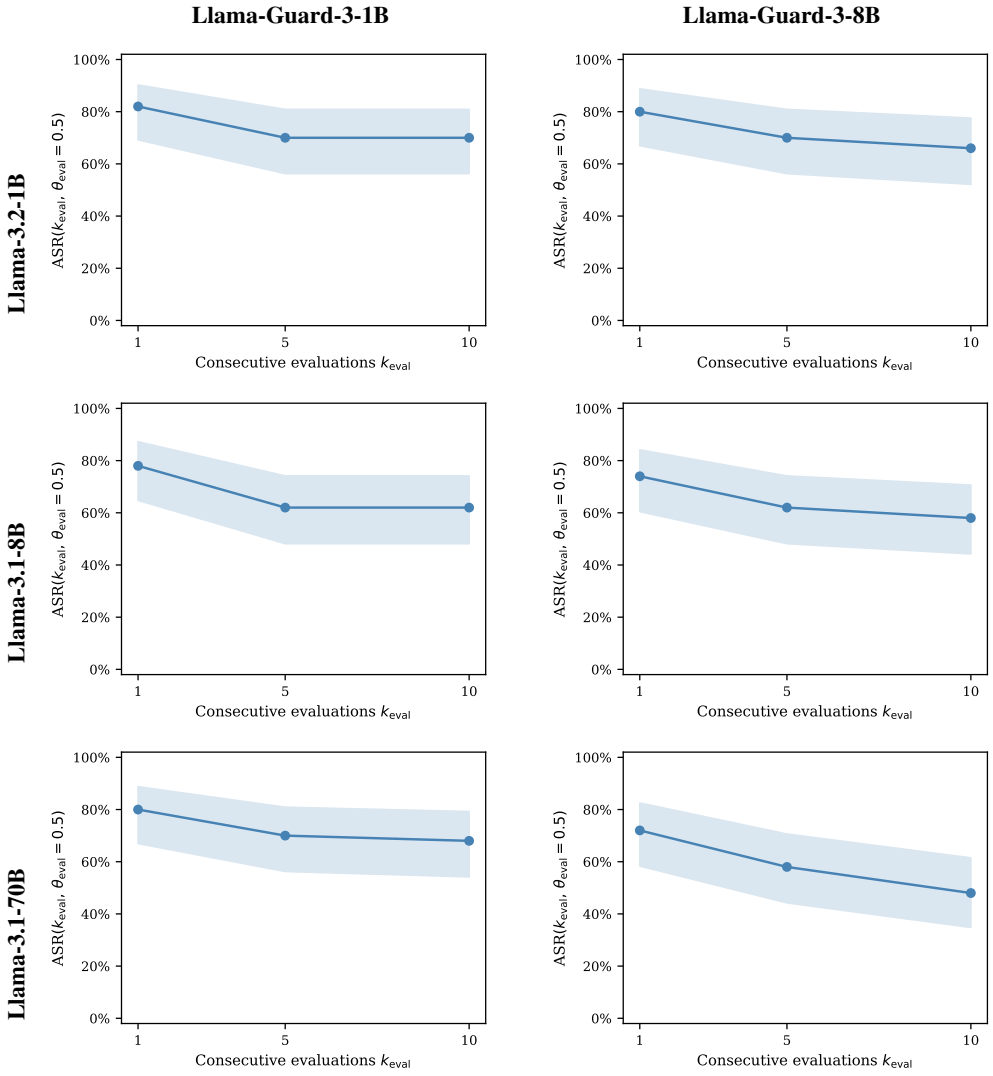


Figure 8: **ASR**(k_{eval}); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5, θ_{eval}=0.5

D.1.2 ASR BASED ON TARGET MODEL TEMPERATURE AT EVALUATION

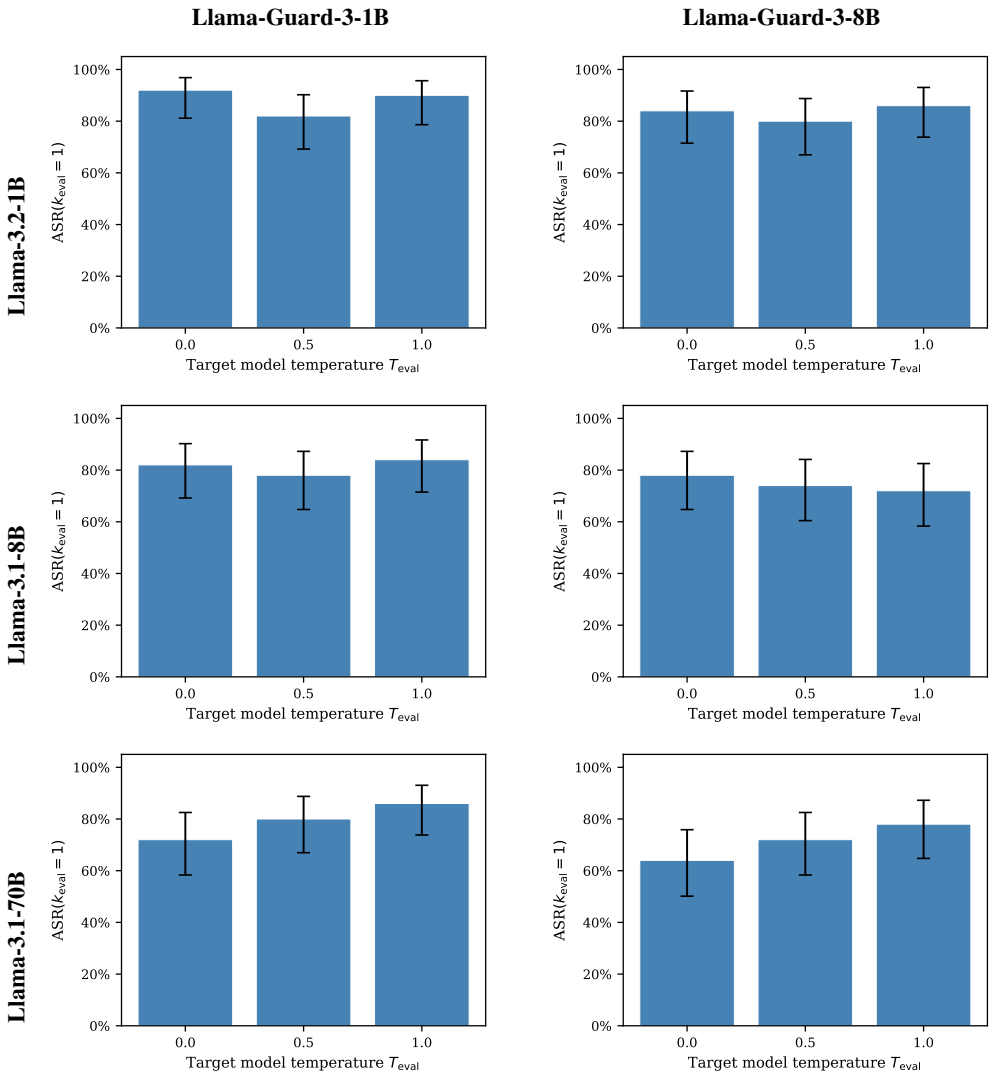


Figure 9: **ASR**($k_{eval} = 1$, T_{eval}); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, θ_{eval}=0.5

D.1.3 ASR BASED ON JUDGE MODEL TEMPERATURE AT EVALUATION

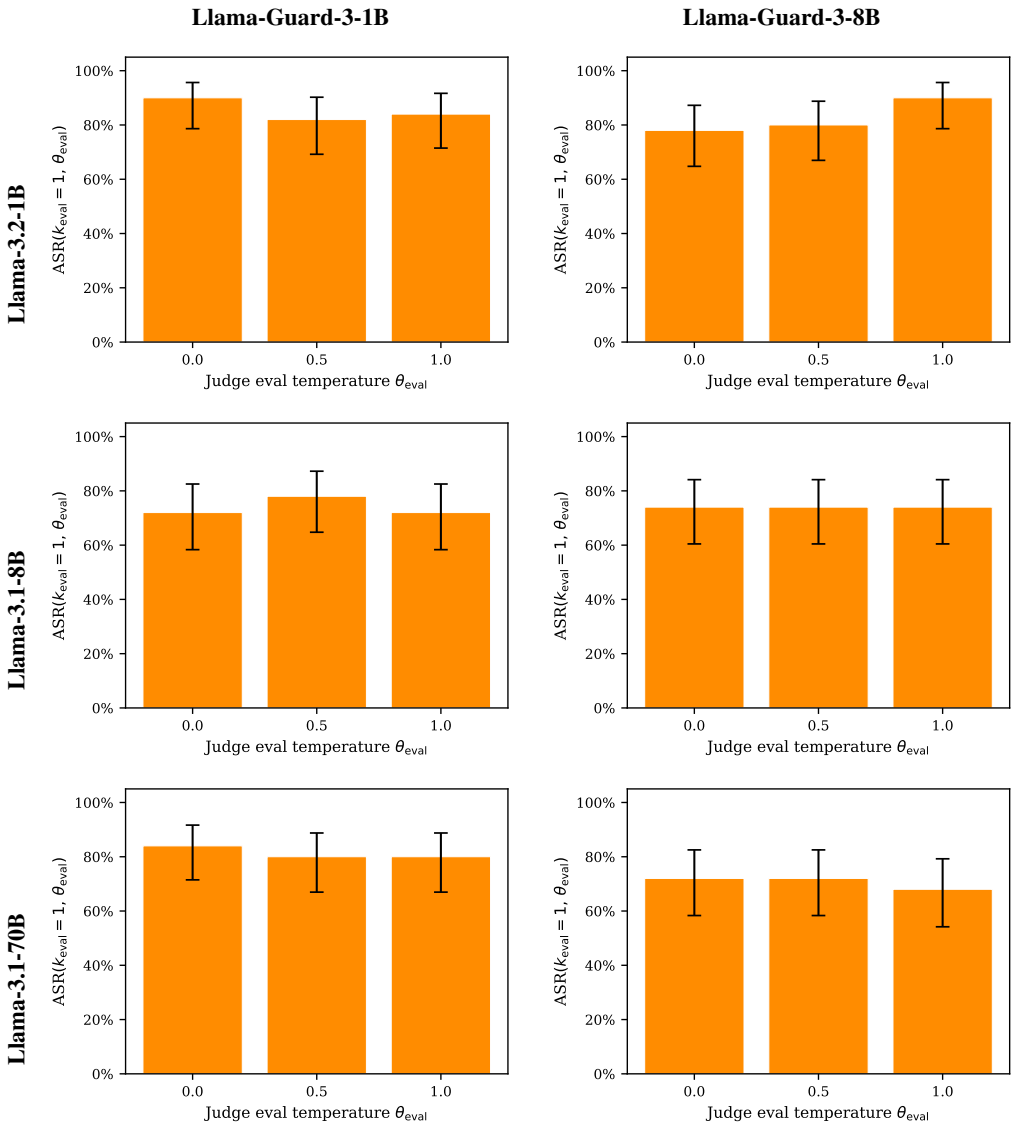


Figure 10: **ASR($k_{eval}=1$, θ_{eval})**; Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5

D.1.4 HEATMAP ASR BASED ON CONSECUTIVE EVALUATIONS AT GENERATION

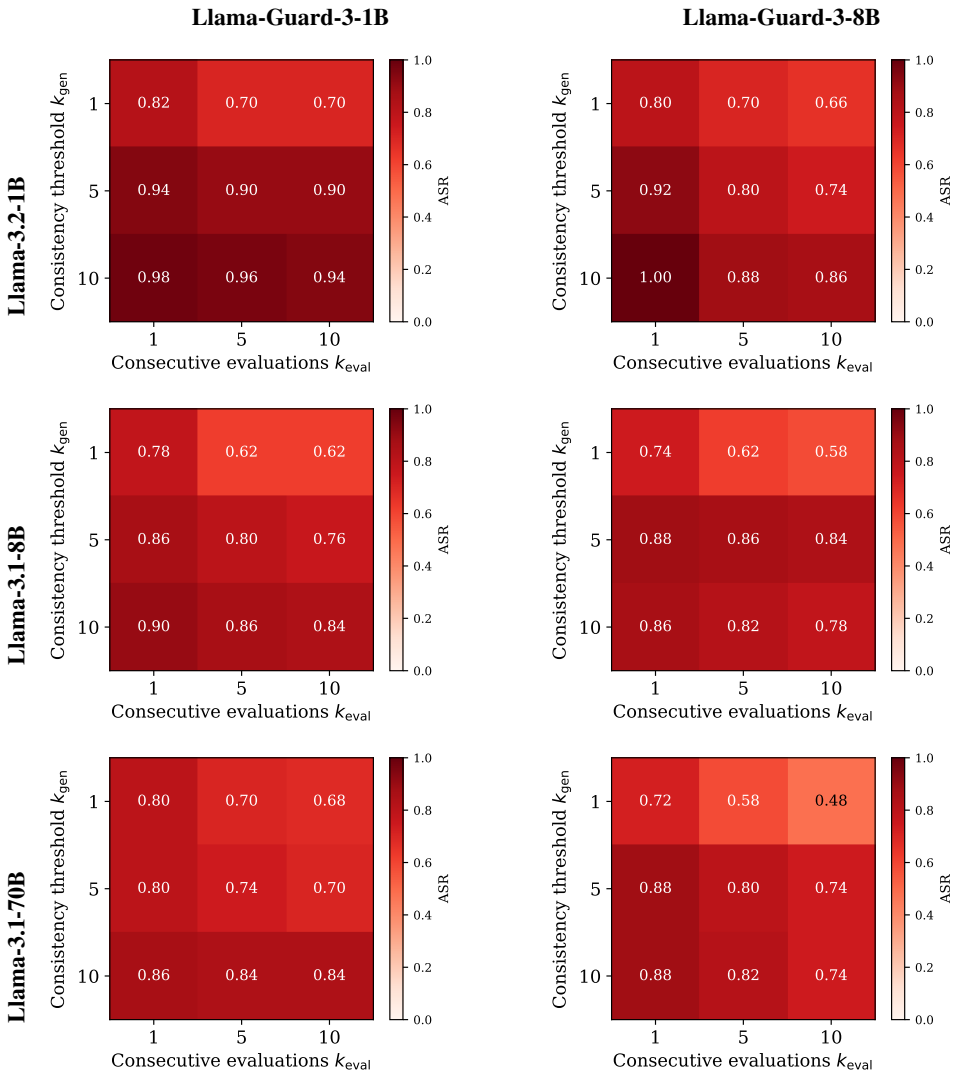


Figure 11: Heatmap ASR(k_{eval}, k_{gen}) ; Parameters: T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5

D.1.5 HEATMAP ASR BASED ON THE TEMPERATURE OF THE TARGET MODEL AT GENERATION

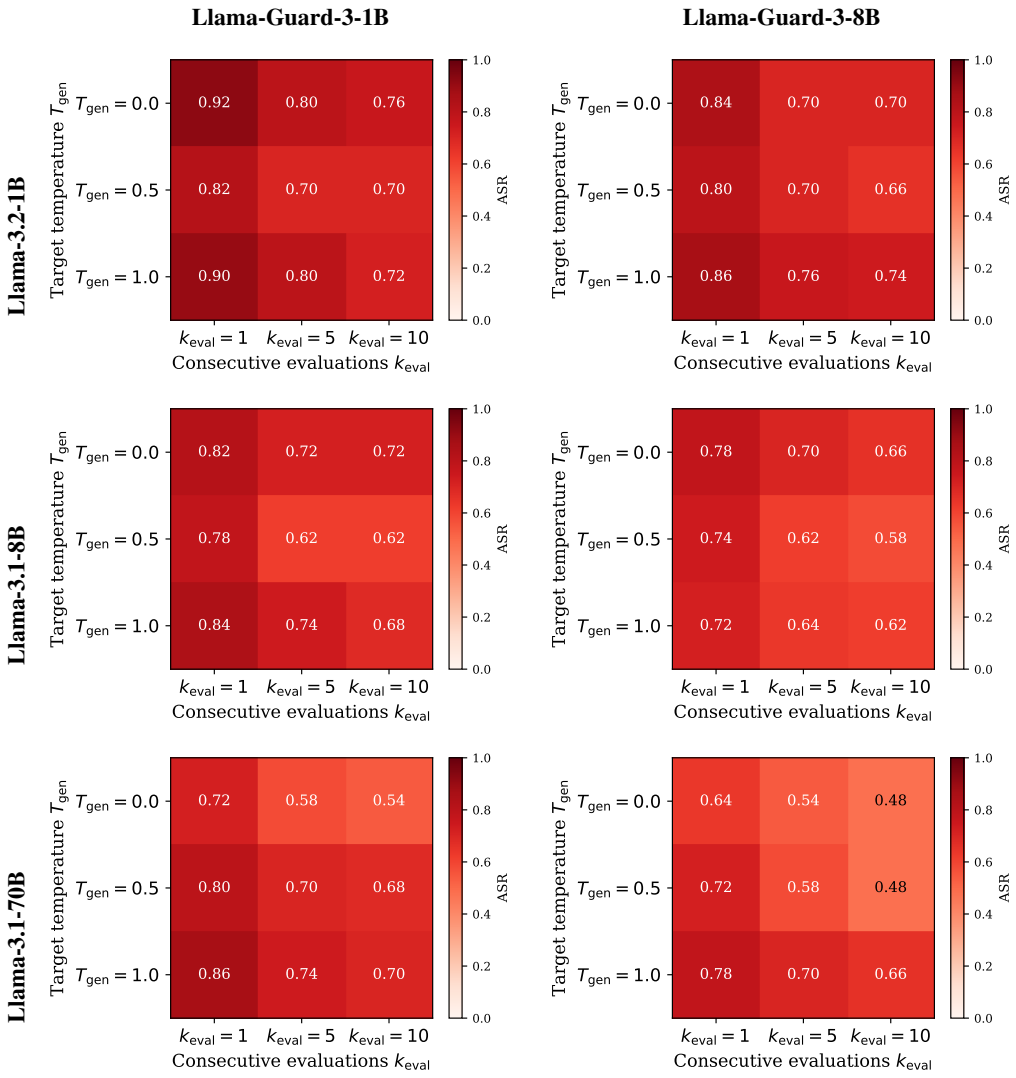


Figure 12: Heatmap ASR(k_{eval}, T_{gen}) ; Parameters: θ_{gen}=0.5, $k_{gen} = 1, T_{\text{eval}}$=0.5, θ_{eval}=0.5

D.1.6 HEATMAP ASR BASED ON THE TEMPERATURE OF THE JUDGE AT GENERATION

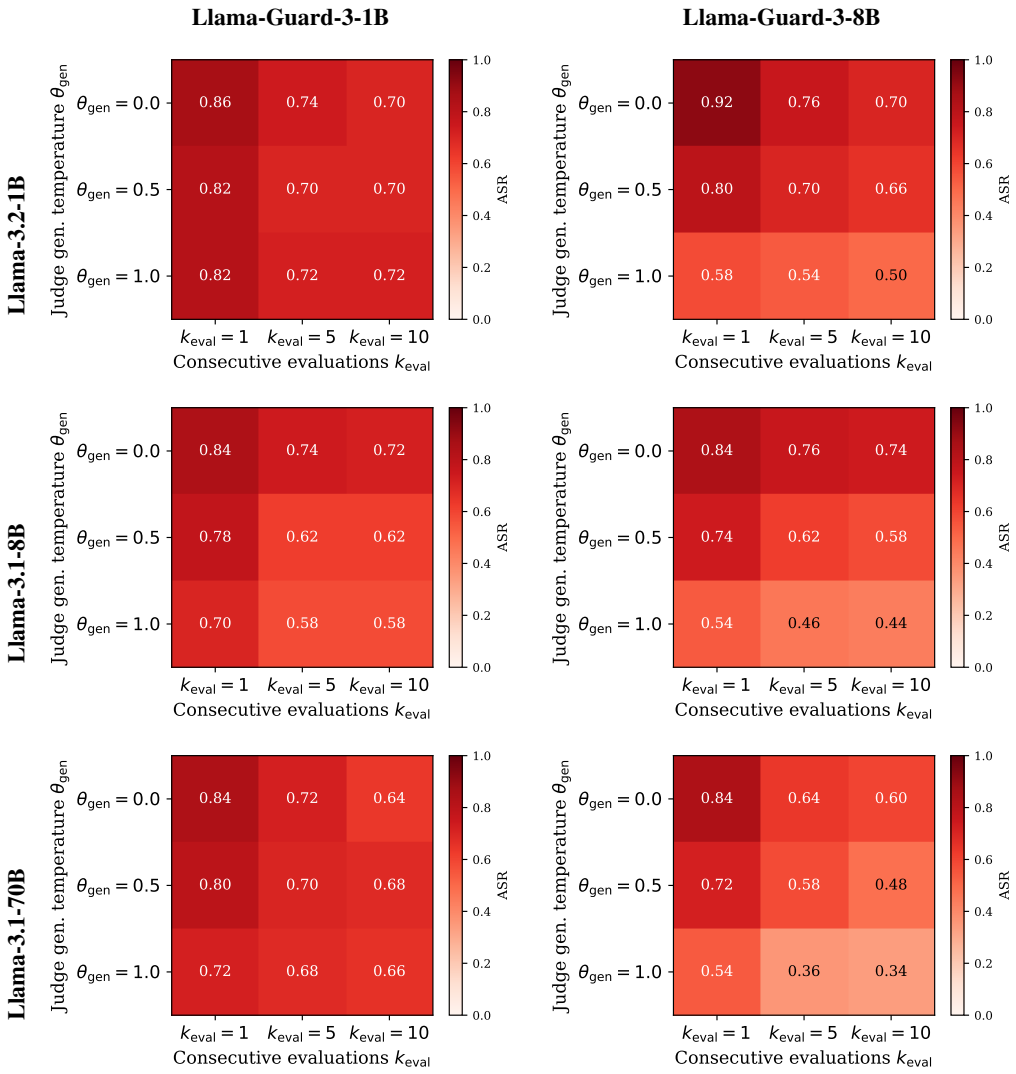


Figure 13: Heatmap ASR($k_{eval}, \theta_{\text{gen}}$) ; Parameters: T_{gen}=0.5, $k_{gen} = 1$, T_{eval}=0.5, θ_{eval}=0.5

D.2 DIFFERENT MODEL PROVIDERS

D.2.1 ASR BASED CONSECUTIVE JUDGE EVALUATIONS

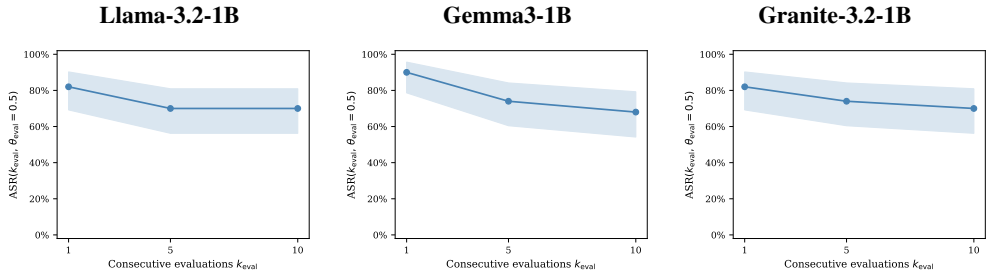


Figure 14: **ASR**(k_{eval}); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5, θ_{eval}=0.5

D.2.2 ASR BASED ON TARGET MODEL TEMPERATURE AT EVALUATION

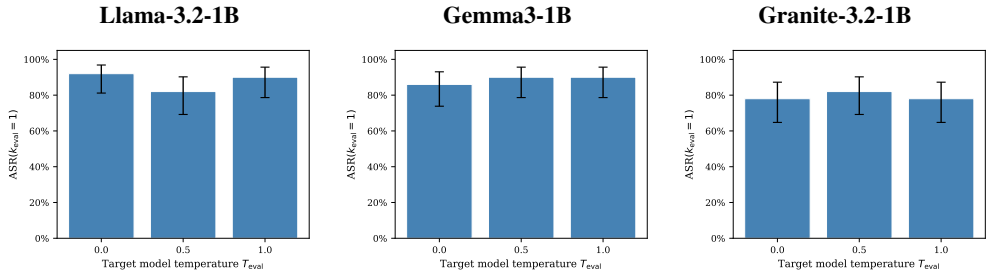


Figure 15: **ASR**($k_{eval} = 1, T_{eval}$); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, θ_{eval}=0.5

D.2.3 ASR BASED ON JUDGE MODEL TEMPERATURE AT EVALUATION

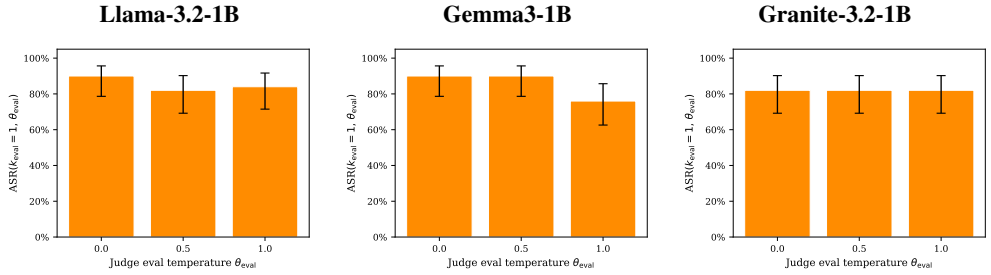


Figure 16: **ASR**($k_{eval} = 1, \theta_{\textbf{eval}}$); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5

D.2.4 HEATMAP ASR BASED ON CONSECUTIVE EVALUATIONS AT GENERATION

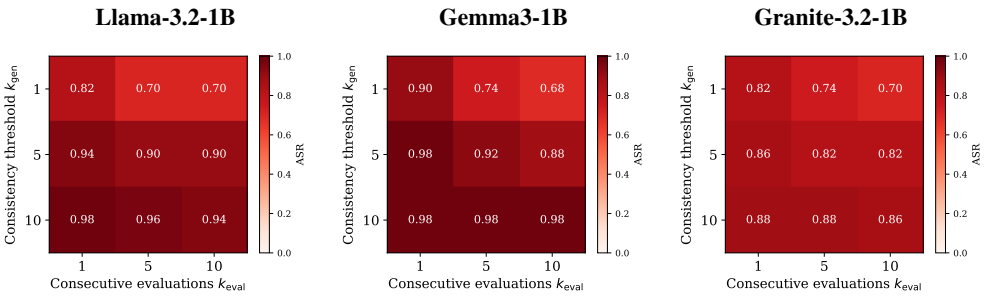


Figure 17: Heatmap ASR(k_{eval}, k_{gen}) ; Parameters: T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5

D.2.5 HEATMAP ASR BASED ON THE TEMPERATURE OF THE TARGET MODEL AT GENERATION

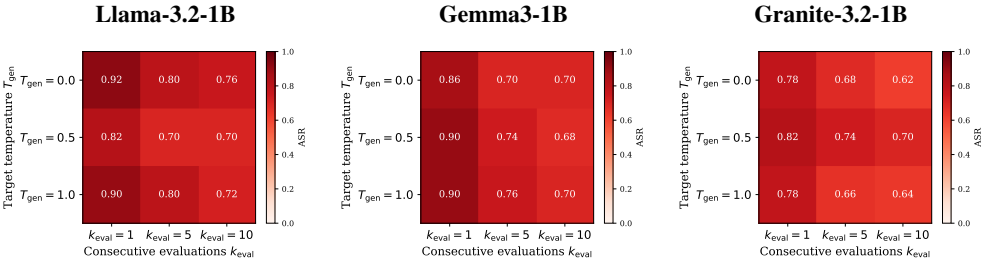


Figure 18: Heatmap ASR(k_{eval}, T_{gen}) ; Parameters: θ_{gen}=0.5, $k_{gen} = 1$, T_{eval}=0.5, θ_{eval}=0.5

D.2.6 HEATMAP ASR BASED ON THE TEMPERATURE OF THE JUDGE AT GENERATION

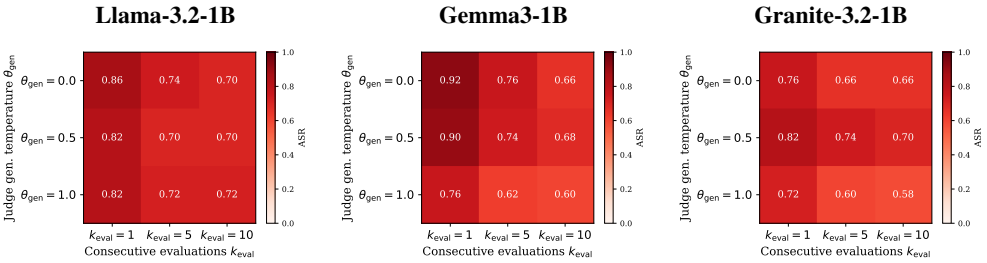


Figure 19: Heatmap ASR($k_{eval}, \theta_{\text{gen}}$) ; Parameters: T_{gen}=0.5, $k_{gen} = 1$, T_{eval}=0.5, θ_{eval}=0.5

D.3 DIFFERENT ATTACKS

D.3.1 ASR BASED CONSECUTIVE JUDGE EVALUATIONS

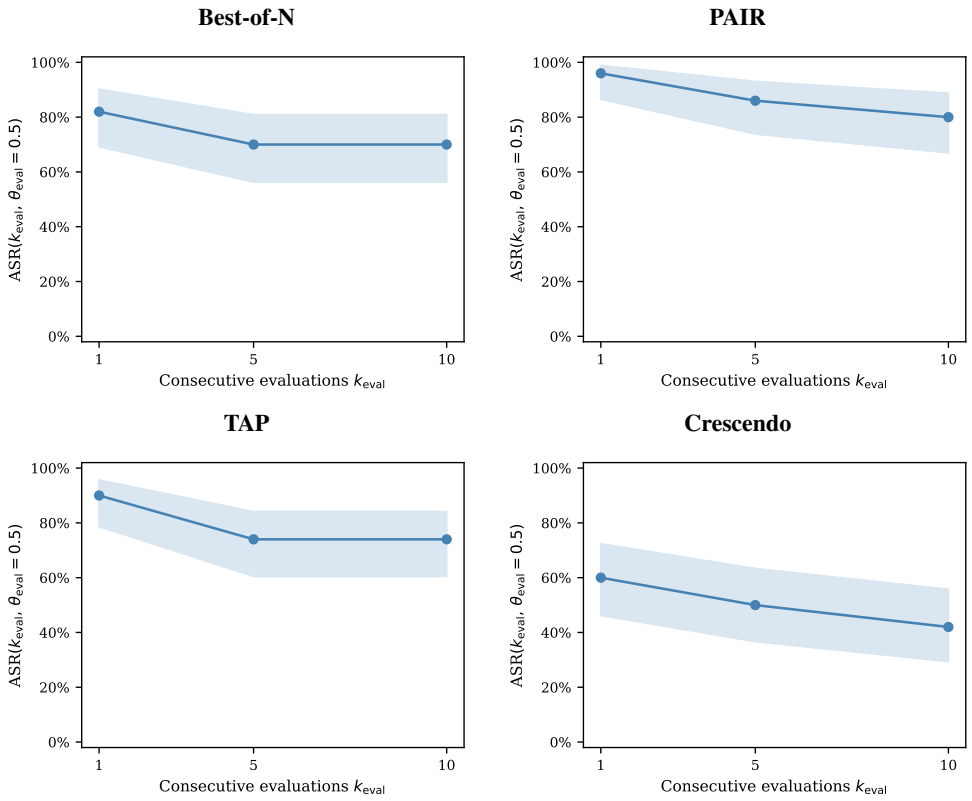


Figure 20: **ASR**(k_{eval}); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5, θ_{eval}=0.5

D.3.2 ASR BASED ON TARGET MODEL TEMPERATURE AT EVALUATION

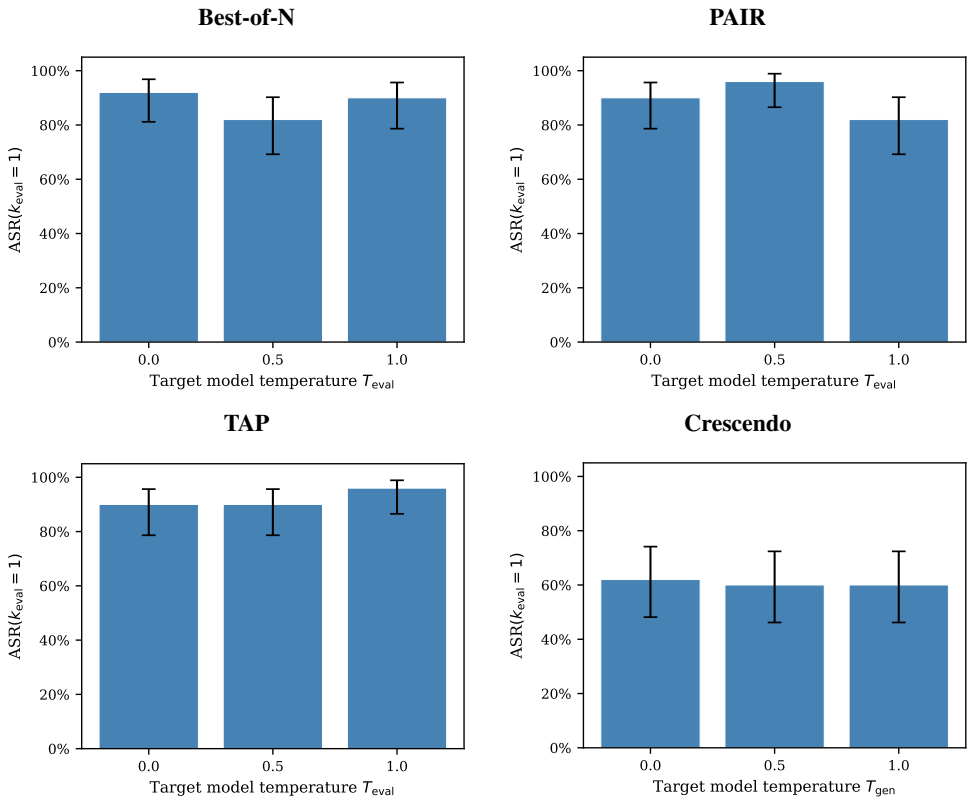


Figure 21: **ASR**($k_{eval} = 1$, T_{eval}); Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, θ_{eval}=0.5

D.3.3 ASR BASED ON JUDGE MODEL TEMPERATURE AT EVALUATION

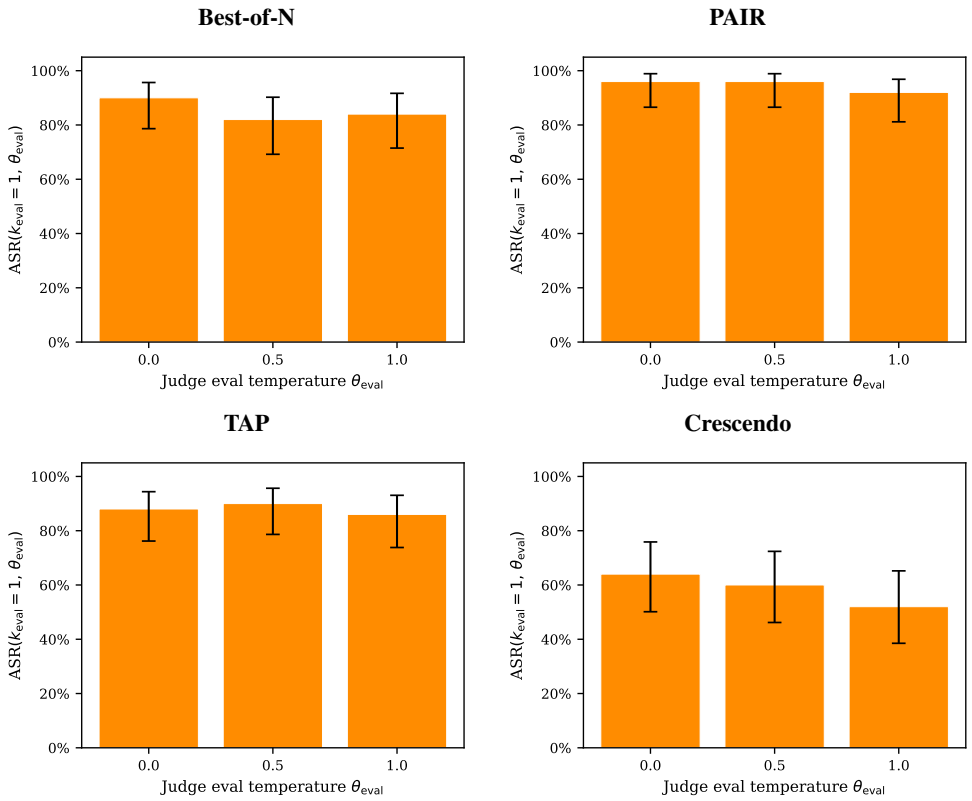


Figure 22: **ASR($k_{eval} = 1$, θ_{eval})**; Parameters: k_{gen}=1, T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5

D.3.4 HEATMAP ASR BASED ON CONSECUTIVE EVALUATIONS AT GENERATION

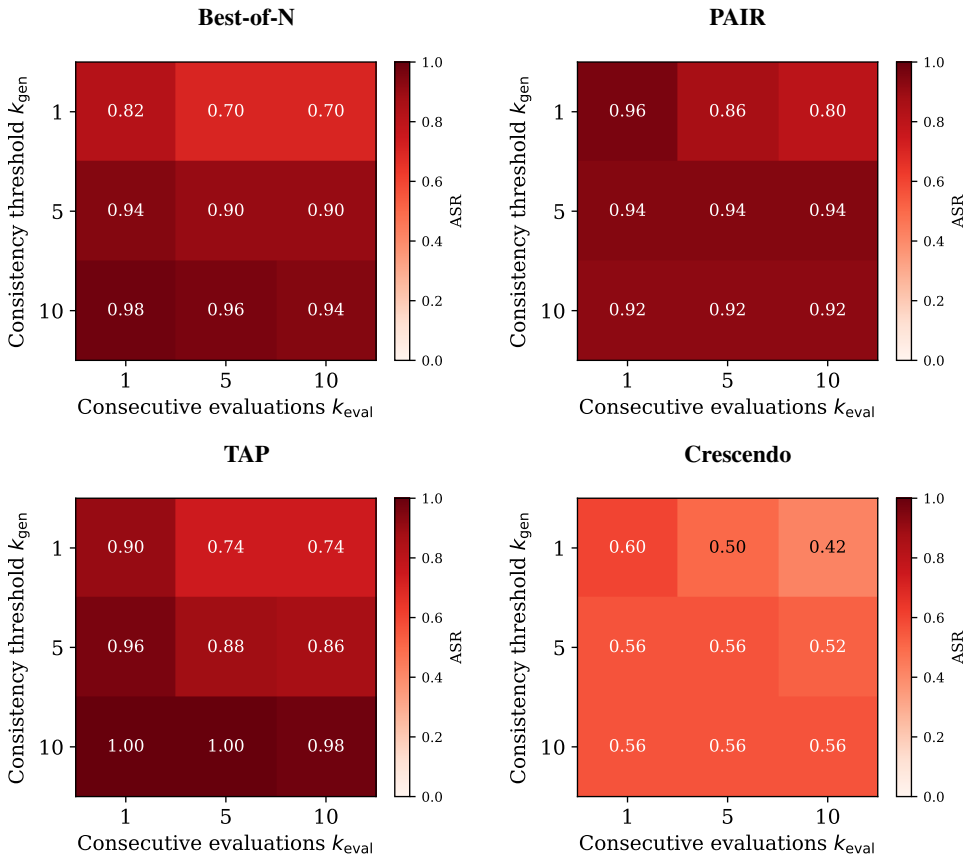


Figure 23: Heatmap ASR(k_{eval}, k_{gen}) ; Parameters: T_{gen}=0.5, θ_{gen}=0.5, T_{eval}=0.5

D.3.5 HEATMAP ASR BASED ON THE TEMPERATURE OF THE TARGET MODEL AT GENERATION

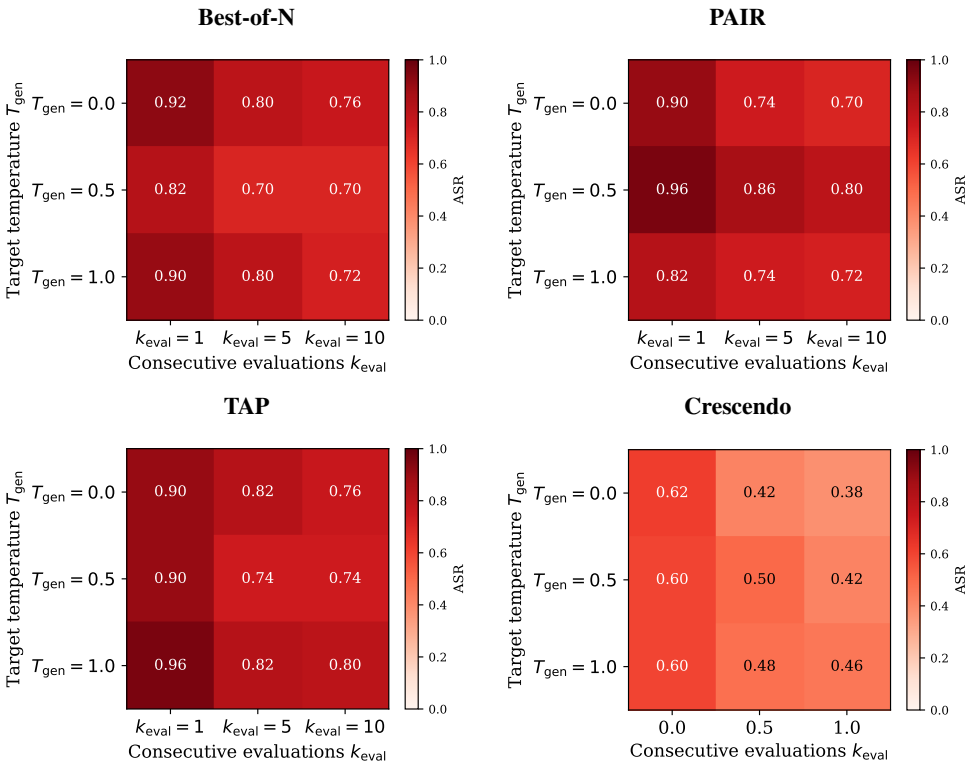


Figure 24: Heatmap ASR(k_{eval}, T_{gen}) ; Parameters: θ_{gen}=0.5, $k_{gen} = 1, T_{\text{eval}}$=0.5, θ_{eval}=0.5

D.3.6 HEATMAP ASR BASED ON THE TEMPERATURE OF THE JUDGE AT GENERATION

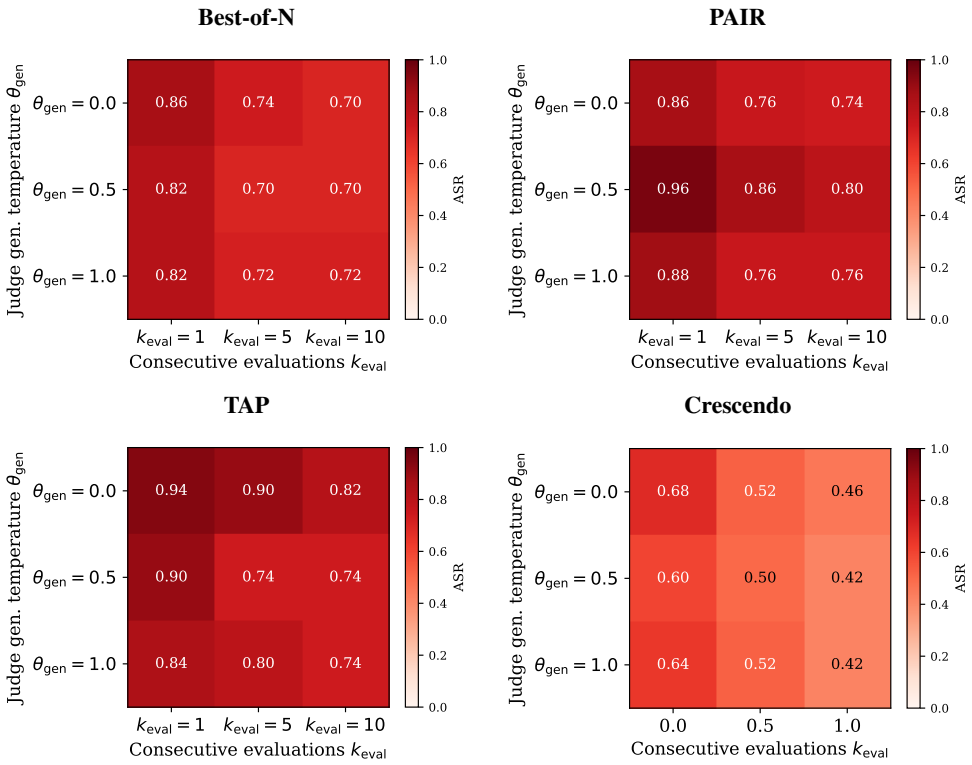


Figure 25: Heatmap ASR($k_{eval}, \theta_{\text{gen}}$) ; Parameters: T_{gen}=0.5, $k_{gen} = 1$, T_{eval}=0.5, θ_{eval}=0.5