A Robust, Defensible, and Reproducible Methodology for Benchmarking Single-Turn Jailbreak Attacks on Large Language Models

◇ Lead Author * Core Authors † Contributing Authors ‡ Supporting Contributors

Carsten Maple[◇1,2,3], Riya Tapwal[*3], Chris Knotz[*4,5], James Ezick[*6], James Goel[6], Alicia Parrish[*7], Federico Ricciuti[*8], Jonathan Petit[*6], Ong Chen Hui[*9], Seok Min Lim[*9], Armstrong Foundjem[*10], Tuesday[†11], Elie Alhajjar[†12], Sean McGregor[†13], Kashyap Ramanandula Manjusha[†14], Andrew Gruen[†15], Bennett Hillenbrand[*5], Jacqueline Stetson[†5], Kurt Bollacker[†5], Abhishek Kumar[*1], Kongtao Chen[†17], Simone Tedeschi[†18], Aakash Gupta[†19], Vasilios Mavroudis[†1], Victor Lu[†8], Rajat Shinde[†20], Satyapriya Krishna[†21], Vincent Alessi[†22,23], Jibin Varghese[†24], Roman Eng[†25], Virendra Mehta[†26], Anita Khadka[†3], Bo Li[†14,27], Washington Mbonu[†3], Saif Ul Islam[†3], Cong Chen[†10], Vassil Tashev[†8], Manish Bhatt[†28], Matteo Prandi[†29], Serrhini Mohammed[†30], Tianhao Li[†31], Sarah Luger[†5,32], Chu, Hua-Rong[†33], Leon Derczynski[†24], Murali emani[†34], Saurav Sahay[†35], Faiza Khan Khattak[†36], Rajat Ghosh[‡16], Nobin Sarwar[‡37], Iris Johnson[‡38], Dhivya Nagasubramanian[‡8], Xuanli He[‡39], Eugenia Kim[‡40], Sujata Goswami[‡41], Fazl Barez[‡42,43], Jean-Philippe Monteuuis[‡6], Sanket Badhe[‡17], Foutse Khomh[‡10], Chee Seng Chan[‡44], Priyanka Tiwari[‡45], Ying-Jung Chen[‡8], Lora Aroyo[‡7], Kenneth Fricklas[‡46], Manil Maskey[‡47], Trupti Bavalatti[‡45], Bryan Tegomoh[‡48], and Natan Vidra[‡49]

[1] *The Alan Turing Institute* — [2] *Verify AI* — [3] *University of Warwick* — [4] *Common Ground* — [5] *MLCommons* — [6] *Qualcomm* — [7] *Google DeepMind* — [8] *Independent* — [9] *IMDA* — [10] *Polytechnique Montreal* — [11] *Artifex* — [12] *RAND* — [13] *AVERI* — [14] *UIUC* — [15] *WorkingPaper* — [16] *Nutanix* — [17] *Google* — [18] *Sapienza University of Rome* — [19] *Think Evolve Labs* — [20] *University of Alabama in Huntsville* — [21] *Amazon Nova* — [22] *University of Michigan* — [23] *University of Utah* — [24] *NVIDIA* — [25] *Clarkson University* — [26] *System Two AI* — [27] *Virtue AI* — [28] *Amazon Leo* — [29] *Dexai srl* — [30] *University Mohamed First Oujda Morocco* — [31] *Duke University* — [32] *iMerit* — [33] *Chunghwa Telecom Laboratories* — [34] *Argonne National Laboratory* — [35] *Intel* — [36] *Monark Health* — [37] *University of Maryland* — [38] *NHLStenden* — [39] *University College London* — [40] *Microsoft* — [41] *Berkeley National Laboratory* — [42] *University of Oxford* — [43] *Martian* — [44] *Universiti Malaya* — [45] *Meta* — [46] *Turaco Strategy* — [47] *NASA* — [48] *University of California, Berkeley* — [49] *Anote*

February 16, 2026

Keywords: AI security evaluation, adversarial testing, jailbreak attacks, resilience gap, MLCommons AILuminate, Multimodal Safety Test Suite, large language models, vision language models, AI governance, ISO/IEC 42001.

Executive Summary

Large language models (LLMs) are deployed in safety-critical, enterprise, and regulated environments where reliability under adversarial conditions is a prerequisite for trust. In these settings, robustness to single-turn inference-time jailbreak attacks, prompt manipulations designed to bypass safety controls, is not merely a research concern but an operational and governance requirement. Prior benchmarking efforts, including AILuminate v1.0 and MLCommons Jailbreak Benchmark v0.5, have established shared infrastructure for measuring harmful outputs and safety degradation under attack. However, as deployment contexts mature and regulatory scrutiny increases, benchmarking must evolve beyond simple collections of adversarial prompts. Organizations require evaluation methodologies that are not only technically sound but also reproducible, interpretable, and defensible to auditors and governance bodies. This paper addresses that need by proposing a taxonomy-driven methodology for benchmarking single-turn jailbreak attacks.

Problem

As LLMs are increasingly deployed in regulated and governance-sensitive environments, jailbreak benchmarking must support not only technical robustness assessment but also defensible and auditable evaluation practices. However, existing jailbreak evaluations exhibit structural limitations that prevent them from meeting these requirements.

- Lack of Defensibility: Many benchmarks rely on a selection of attacks without a principled rationale for inclusion. In the absence of a formal taxonomy, it remains unclear why certain attacks are selected and others omitted, coverage claims cannot be systematically justified, and attack selection appears discretionary rather than methodologically grounded. Consequently, external auditors cannot verify whether the benchmark meaningfully represents the attack surface, undermining governance credibility.
- Coverage Gaps Across Attack Families: Without structured classification, widely publicized jailbreak styles dominate evaluation corpora, while less visible but structurally distinct mechanisms—such as encoding-based obfuscation, semantic-preserving perturbations, and compositional instruction strategies may be underrepresented. This imbalance risks overfitting to familiar attack patterns, producing incomplete adversarial coverage and misleading robustness estimates. Given the combinatorial space of prompt manipulations, balanced sampling across mechanisms is not achievable without an explicit structural framework.
- Reproducibility and Verifiability: Reproducibility is compromised when categories overlap, operate at inconsistent abstraction levels, or lack deterministic placement rules. Without one-instance-to-one-category assignment, inter-annotator agreement weakens and cross-organizational comparisons become unstable. Furthermore, verifiability requires clear scope boundaries, explicit decision rules, executable category definitions, and transparent corpus construction. When categories remain abstract or aggregate heterogeneous mechanisms, independent validation and mechanism-level diagnosis become difficult, limiting both auditability and longitudinal interpretability.

Contribution

In response to the structural shortcomings identified above, this paper introduces a mechanism-first, benchmark-operational taxonomy of single-turn jailbreak attacks together with a structured evaluation methodology. The taxonomy is designed not as a descriptive catalog, but as a foundational component of benchmark construction, governing attack selection, corpus design, labeling, and reporting.

- Mechanism-First Taxonomy: Rather than organizing attacks by the hazards they trigger, the taxonomy classifies them according to how prompts manipulate model behavior at inference time. This mechanism-level organization enables stratified robustness reporting, clearer attribution of failure modes, and balanced sampling across heterogeneous bypass strategies. To ensure reproducibility, each prompt maps deterministically to exactly one leaf category, enforcing a strict one-instance-to-one-leaf constraint.
- Evaluation-Driven Design Requirements: The taxonomy satisfies six requirements necessary for benchmark operation: prompt-only scope, mechanism-first organization, deterministic instance assignment, consistent splitting rules, executable category definitions, and coverage across major bypass families. These constraints ensure that classification supports reproducible corpus construction, transparent inclusion criteria, and interpretable mechanism-level metrics.
- Structured Development and Tiered Selection: The taxonomy is constructed through a formal, multi-stage development process that defines scope boundaries, establishes a governing meta-characteristic, and iteratively refines category structure to ensure stability over time. The paper introduces a tiered attack-selection strategy: Tier 1 mechanisms support benchmark reporting, while additional tiers (Tier 2 and Tier 3) provide broader research coverage. This balances comprehensiveness with reproducibility and controlled corpus growth.

Methodological Lessons

Building on the mechanism-first taxonomy, the benchmark construction process is structured to ensure defensible attack selection, balanced coverage, and reproducible evaluation across systems and releases.

Benchmark establishment should proceed through a structured sequence of steps: first, attacks should be selected using taxonomy-guided sampling to ensure balanced coverage across mechanism families and deterministic instance placement; second, executable prompt transformations should be implemented with documented and parameter-controlled generation procedures to support reproducibility and auditability; third, multiple variants per mechanism should be instantiated where necessary to capture structural diversity without altering category semantics; fourth, systems should be evaluated under a paired protocol in which baseline performance on seed prompts is measured prior to re-evaluation under adversarially transformed inputs; and finally, results should be analyzed using mechanism-stratified reporting and attack-aware reliability assessment to distinguish model robustness from evaluator variability and to preserve longitudinal comparability across benchmark releases.

Significance

The paper's central insight is that credible jailbreak benchmarking requires structural discipline at the taxonomy level. Without a defensible classification backbone, improvements in robustness cannot be confidently interpreted, compared across systems, or justified to regulators.

By linking taxonomy construction directly to corpus design and evaluation methodology, the work shifts jailbreak benchmarking from informal red-teaming toward reproducible, mechanism-stratified, governance-aligned security evaluation.

Contents

1 Introduction

Large language models (LLMs) are now widely deployed in enterprise systems, consumer-facing applications, and regulated environments such as healthcare, finance, and public-sector decision support [1]. In these settings, safety and reliability are no longer abstract research concerns but operational requirements tied to governance processes, regulatory scrutiny, and organizational risk management. Industry benchmarking efforts such as AILuminate v1.1 and the MLCommons Jailbreak Benchmark v0.5 reflect this shift by providing shared evaluation infrastructure for measuring harmful system behavior and susceptibility to adversarial prompting [2, 3].

A growing body of academic and practitioner work demonstrates that even safety-aligned models remain vulnerable to *jailbreaks*: inference-time prompt attacks that induce unsafe or policy-violating behavior without requiring access to model weights, training data, or system internals [4, 5]. These attacks exploit properties of natural-language interfaces and prompt-processing pipelines, making them directly relevant to deployed systems rather than hypothetical threat models [6]. As a result, such vulnerabilities persist even when models have been fine-tuned or fitted with conventional safety filters, challenging assumptions about the robustness and operational safety assurance of LLM-based services in real-world workflows.

Susceptibility to jailbreak attacks has implications that extend beyond technical performance metrics [7, 6]. For organizations subject to emerging AI governance regimes, failures under adversarial prompting complicate internal assurance processes, audit readiness, and regulatory reporting [8, 9, 7]. When adversarial testing lacks principled structure, it becomes difficult to justify coverage claims, to explain observed failures in mechanism-level terms, or to demonstrate that mitigation efforts address stable classes of attack rather than isolated prompt templates [10, 4, 11, 12]. Taken together, these pressures make ad hoc red-teaming and isolated case studies insufficient for deployment assurance. Organizations increasingly require systematic benchmarking frameworks that can quantify robustness under attack, support comparisons across models and releases, and generate evidence suitable for internal risk management, audit processes, and regulatory engagement. In contrast to informal attack collections or one-off evaluations, security benchmarks provide a common experimental apparatus: they define scope boundaries and threat models, specify how attacks are selected and instantiated, prescribe evaluation protocols, and standardize reporting formats. Without such shared structure, observed failures are difficult to contextualize, coverage claims remain ambiguous, and improvements over time cannot be distinguished from shifts in the tested attack set. However, the mere existence of a benchmark is not sufficient. To support deployment decisions, regulatory scrutiny, and longitudinal safety claims, jailbreak benchmarks must satisfy strong methodological properties. In particular, evaluations must be robust to variation in attack realizations, repeatable and reproducible across organizations and experimental runs, and grounded in design choices that are defensible to external auditors and governance bodies. For security assessment, defensibility requires that scope boundaries are explicit, threat models are justified, attack selection procedures are principled rather than ad hoc, and coverage claims can be explained in mechanism-level terms rather than by reference to opaque prompt collections. Justification and evidence collection has gained increasing traction in assurance of AI systems, see [13], [14], [15], though this has not widely been adopted in benchmarking practices.

This work responds to these requirements by proposing a methodology for benchmarking single-turn jailbreak attacks that emphasizes robustness, reproducibility, and defensibility from first principles.

Rather than prioritizing dataset scale or leaderboard breadth, the approach centers on a taxonomy-driven organization of prompt-based attack mechanisms, coupled with verified implementations and principled sampling strategies. Meaningful jailbreak evaluation, we argue, depends not merely on the number of prompts applied, but on comprehensive coverage across distinct attack classes and on stress-testing systems under systematically varied adversarial conditions. By grounding evaluation in an explicit mechanism-level structure, the methodology aims to provide a stable foundation for large-scale benchmarks capable of supporting credible robustness claims and comparative safety assessment.

1.1 Limitations of Existing Jailbreak Evaluations

Earlier releases, including MLCommons Jailbreak Benchmark v0.5 and the AILuminate safety benchmark, made two foundational contributions to large-scale safety evaluation. First, they established standardized adversarial testing workflows in which models were evaluated under adversarial prompting conditions using shared infrastructure and grading rubrics [3, 2]. Second, they relied on common evaluators and calibrated grade bands, enabling the reporting of safety degradation in a manner comparable across organizations and model versions. Parallel academic and practitioner work likewise developed prompt-based red teaming pipelines and cataloged jailbreak behaviors across contemporary models [16, 17, 6]. Surveys and systematization efforts further clarified the diversity of prompt manipulation techniques, ranging from role playing and obfuscation to multi-step coercion and indirect prompt injection [4].

Experience with these evaluations, however, revealed structural limitations that constrained comprehensiveness, verifiability and long-term reproducibility. In v0.5, adversarial prompts were largely grouped using descriptive or ad hoc labels rather than a systematically defined mechanism-level taxonomy. AILuminate, by design, focused on classifying hazardous *outcomes* rather than adversarial *methods*, leaving open questions about how different prompt-manipulation strategies relate to observed failures [18]. As a result, benchmark users encountered difficulty answering questions such as:

- which classes of prompt-manipulation mechanisms a model resists or succumbs to;
- whether failures observed under one hazard category generalize across others when induced through similar attack strategies.

The absence of a formal taxonomy also introduces sampling risks that are well documented in the adversarial-ML literature [19, 20, 21]. Widely publicized jailbreak styles, such as role-play templates and persona-based overrides, tend to dominate informal attack collections, while less visible but technically distinct manipulation strategies remain underrepresented [4, 5]. Such skew complicates coverage claims. It also becomes difficult to tell whether a system is truly more robust, or if the results just changed because the set of attacks being tested is different.

More broadly, the absence of an explicit classification scheme undermines several properties that are central to the credibility of security evaluations for LLMs:

- **Robustness and comprehensiveness:** Without mechanism-level categories, evaluations often concentrate on a narrow set of prompt templates while omitting structurally distinct attack families, leading to misleading conclusions about system resilience and an incomplete assessment of adversarial coverage.

- **Reproducibility:** When different organizations construct attack corpora under the same informal labels, cross-study comparability deteriorates and longitudinal analysis becomes unreliable.
- **Verifiability and defensibility:** Poorly specified categories and ad hoc corpus construction make it difficult to confirm that prompts belong to the claimed class, to justify coverage claims, or to explain methodological choices to auditors, regulators, and other external stakeholders. From a methodological and research perspective, an explicit taxonomy further contributes to defensibility by making the structure of the evaluated attack surface transparent: it delineates the range of mechanisms considered in scope, clarifies which classes are intentionally excluded, and reveals the relative breadth and depth of coverage across different manipulation families. Such visibility enables researchers to reason systematically about what conclusions a given evaluation framework does and does not support, to identify gaps in existing testing regimes, and to use the classification structure as a scaffold for extending empirical studies and developing new defensive techniques.

These limitations motivate the main methodological shift advanced in this work: rather than treating attack organization as a secondary descriptive step, we argue that taxonomy design must be a central decision in any evaluation methodology for single-turn jailbreak attacks, governing how attacks are selected, instantiated, and reported. The remainder of the paper shows how these properties: robustness, reproducibility, and defensibility are operationalized through explicit scope boundaries, deterministic classification rules, principled sampling strategies, and transparent reporting.

1.2 Rationale for a Taxonomy-Driven Evaluation Methodology

In this paper, we prioritize the development of an explicit, benchmark-operational taxonomy over immediate expansion of datasets, model coverage, or leaderboard-style reporting. Meaningful and durable jailbreak evaluation depends first on defensible classification decisions that govern what is tested, how attacks are sampled, and how results are aggregated [10].

Benchmark results are inseparable from the structure used to organize adversarial prompts [22, 23]. Any jailbreak evaluation implicitly commits to a set of classification rules that determine:

- which classes of attacks are included in scope;
- how individual prompt instances are sampled within those classes;
- how results are aggregated across heterogeneous attack families;
- and how coverage claims are justified to external users.

When such decisions remain informal or underspecified, several methodological problems arise. Coverage statements become difficult to substantiate because the space of possible attack mechanisms is not clearly partitioned. Apparent improvements in model robustness may reflect changes in the composition of the evaluated attack set rather than genuine advances in defensive capability. Longitudinal and cross-organizational comparisons likewise weaken when successive releases draw on different, implicitly defined notions of what constitutes a jailbreak, concerns that also surface in recent surveys of prompt attacks and adversarial prompting [4, 5].

For these reasons, we treat taxonomy construction as a methodological objective rather than an auxiliary descriptive exercise. The taxonomy is designed to satisfy three benchmark-critical properties:

- **Mechanism-first organization:** attacks are grouped according to how the prompt manipulates the model at inference time, rather than by the hazards they induce or the policy domains they target;
- **Instance-level assignment:** every concrete prompt instance maps to exactly one leaf category, enabling deterministic labeling, corpus construction, and interpretable statistics, in line with formal taxonomy-development methodologies [24];
- **Benchmark-oriented design:** categories are defined to support executable prompt instances, balanced sampling strategies, and reproducible evaluation, rather than serving solely as high-level conceptual groupings.

1.3 Objectives

Specifically, this paper aims to:

1. establish a benchmark-operational taxonomy of single-turn, inference-time prompt attacks, suitable for deterministic labeling, mechanism-stratified reporting, and longitudinal comparison;
2. define a transparent and robust process for taxonomy development and maintenance, grounded in systematic evidence collection and iterative refinement;
3. demonstrate how a mechanism-first taxonomy structures jailbreak testing, improves the interpretability of results, and enables balanced and reproducible evaluation across attack families;
4. conduct taxonomy-guided adversarial evaluations using representative single-turn attacks and derive methodological lessons that inform future jailbreak benchmarking and assessment efforts;
5. prepare the methodological groundwork for a subsequent MLCommons Jailbreak Benchmark v1.0 that expands coverage across the full taxonomy using reproducible evaluations and audited code artifacts.

In the rest of this paper, the following are described:

1. the background safety benchmarks and prior MLCommons jailbreak releases that motivate the present work;
2. the scope boundaries and threat model assumed throughout the evaluation;
3. the design requirements, development process, and structural principles of the taxonomy;
4. the resulting attack families, splitting rules, and prevalence statistics;
5. the tiered strategy for attack selection and prompt instantiation, together with known gaps and planned extensions.

1.4 Scope of the Proposed Taxonomy

Prior large-scale safety evaluations, such as AILuminate v1.1 and the MLCommons Jailbreak Benchmark v0.5, focus on single-turn, inference-time interactions initiated from English seed prompts, providing standardized safety baselines and adversarial testing for text-based large language models. Building on this foundation, the present work adopts the same single-turn, prompt-only evaluation setting and black-box threat model, while proposing a taxonomy-first evaluation methodology intended to improve reproducibility across systems, runs, and organizations.

The scope of the taxonomy considered in this work is restricted to prompt-only attack mechanisms in which a non-privileged external user attempts to bypass deployed safety or policy constraints through manipulation of a single input. Included classes encompass direct policy-override attempts, semantic-preserving perturbations, encoding and formatting manipulations, and composite single-turn instruction strategies. The methodology does not address multi-turn or conversational manipulation, multilingual or cross-lingual attacks, training-time or infrastructure-level vulnerabilities, indirect prompt injection through tools or retrieval-augmented generation pipelines, or attacks that rely primarily on social engineering. These exclusions are intentional and reflect a design choice to prioritize controlled, reproducible evaluation of single-turn prompt-based mechanisms, while leaving broader threat surfaces for future methodological extensions.

1.5 Contribution

We make several contributions that differentiate this work from prior jailbreak evaluations.

- **Taxonomy-First Benchmark Design:** We introduce an explicit, mechanism-level taxonomy of single-turn prompt attacks and treat classification as a first-order methodological commitment rather than as a descriptive afterthought. The taxonomy defines how attacks are grouped, how boundaries between categories are drawn, and how individual instances are assigned to unique leaves. These structural decisions will determine which attacks are included in the benchmark corpus, how sampling occurs within each family, and how evaluation results are stratified and reported.
- **Mechanism-Level Focus:** Attacks are organized according to how prompts manipulate the model at inference time, rather than by the hazards they trigger or the outcomes they produce. This mechanism-first orientation enables interpretation of evaluation results in terms of specific bypass strategies and supports longitudinal comparisons as defenses evolve.
- **Application-Informed Methodological Lessons:** This work examines attack mechanisms drawn from the proposed taxonomy to derive methodological insights into attack selection, coverage trade-offs, failure-mode characterization, and evaluation design. The emphasis is on articulating generalizable guidance for future single-turn jailbreak evaluation methodologies.
- **Tiered Attack Selection Strategy:** Because the space of possible prompt instantiations within each attack family is combinatorially large, we introduce a tiered approach to attack selection. The benchmark designates a representative subset of mechanisms, referred to as Tier–1 attacks, across which results will be reported in subsequent releases of the MLCommons Jailbreak benchmark. Selection is guided by the taxonomy and by evidence from the academic and practitioner literature, with the objective of balancing breadth across manipulation families while maintaining high implementation quality.

1.6 Structure of the Paper

This paper develops a taxonomy-driven methodology for jailbreak evaluation and applies it within the MLCommons benchmarking framework. The following sections proceed from background benchmarks and scope definition to taxonomy construction, evaluation design, and empirical analysis. Further, Section 2 reviews AILuminate and prior jailbreak releases. Section 3 defines the system model, threat assumptions, scope boundaries, and terminology. Sections 4 and 5 formalize the benchmarking constraints and introduce the taxonomy, including its development process, structural principles, tiered attack-selection strategy, and known gaps. Section 7 describes the multimodal attack frameworks and cultural nuance in Vision-Language Models (VLMs). Section 6 describes the process of establishing benchmarks and their evaluation methodology. Sections 8 and 9 outline future directions and conclude.

1.7 MLCommons Partnership with IMDA

MLCommons has partnered with Infocomm Media Development Authority (IMDA) which leads Singapore's digital transformation by developing a vibrant digital economy and an inclusive digital society. As the Architects of Singapore's Digital Future, IMDA fosters growth in the infocomm technology and media sectors by harnessing frontier technologies and developing the digital infrastructure and talent ecosystems necessary to establish Singapore as a global digital metropolis.

Central to this mission is IMDA's commitment to propelling emerging technologies from lab to reality. The authority drives the development of frontier domains from concept to real-world use by building early and deeply to uncover practical value and identify how new technologies can best benefit industries. By implementing secure digital infrastructure and AI applications, IMDA ensures that innovation can proliferate safely and be adopted at scale.

In the field of Artificial Intelligence, IMDA leads the development of safe and sustainable solutions through a comprehensive governance approach that emphasizes rigorous testing and reliability. To facilitate responsible innovation, the authority has developed open-source technical enablers, such as the AI Verify Toolkit and Project Moonshot, to assess both traditional and generative AI systems. Through these expert-led initiatives and international partnerships, IMDA has established Singapore as a recognized leader in the global AI safety discourse, including the development of culturally-grounded AI benchmarks in collaboration with MLCommons.

1.8 Responsible publication

Since jailbreak prompts can be repurposed for misuse, we avoid publishing verbatim prompt templates or transformation parameters that materially increase attack capability. Instead, we describe attack mechanisms at a structural level and recommend release benchmark artifacts under access controls appropriate to the intended evaluation use case. This approach preserves scientific reproducibility for qualified auditors and benchmarking participants while reducing the risk of providing operational instructions to malicious actors.

2 Background

2.1 The AILuminate AI Safety Benchmark

AILuminate is an industry-driven benchmark designed to support systematic and reproducible evaluation of safety and reliability in deployed LLM systems [2]. It operationalizes safety assessment through standardized prompt sets, graded-response evaluation, and a hazard taxonomy intended to capture a broad range of harmful or policy-violating outputs.

2.1.1 Evaluation Focus

AILuminate focuses on *response behavior* rather than user intent. Given a fixed input, the benchmark evaluates whether a system produces content that violates safety policies or exposes users to harm, regardless of whether the prompt is benign or adversarial. This separation enables AILuminate to serve as a benchmarking framework for jailbreak studies. In our setup, jailbreak attacks are applied to the original seed prompts while preserving their hazard labels. We then evaluate the resulting system responses using the same evaluator ensemble, reference system, and grading procedure defined in AILuminate v1.1.

2.1.2 Hazard Taxonomy

The AILuminate hazard taxonomy organizes unsafe outcomes into several broad families. *Physical hazards* capture responses that meaningfully facilitate bodily harm or dangerous real-world actions. *Non-physical hazards* include illegal activity facilitation, harassment, extremism, and privacy violations. *Contextual and specialized advice hazards* address situations in which the appropriateness of a response depends on user vulnerability or professional context, such as medical, legal, or financial guidance delivered without appropriate safeguards [2]. These top-level families are further subdivided into a set of specific hazard categories, such as violent crimes, non-violent crimes, suicide and self-harm, privacy, hate, sexual content, and specialized advice, that constitute the unit of scoring and reporting in the benchmark. The analyses in this work, therefore, rely on those lower-level categories rather than only the coarse family-level groupings.

2.1.3 Hazards versus Attacks

Throughout this work, a distinction is maintained between hazard taxonomies and attack taxonomies. Based on a hazard taxonomy, AILuminate evaluates potentially harmful outputs in order to support consistent grading of system behavior once a response is produced. By contrast, the taxonomy introduced in this work classifies the mechanisms by which attacked prompts attempt to induce harmful behavior. Therefore, hazard categories describe *what* kind of potential harm can occur, whereas attack categories describe *how* an adversary seeks to create prompts to cause that harm. The two dimensions remain complementary and orthogonal.

2.1.4 Role as a Safety Baseline

AILuminate v1.1 is adopted as the reference benchmark for releases of the MLCommons Jailbreak Benchmark because its hazard definitions reflect a mature, consensus-driven development process

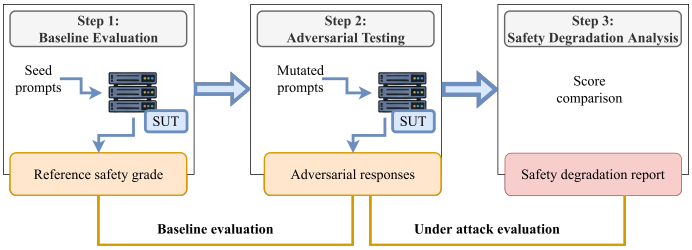

Figure 1: Three-stage evaluation pipeline for measuring robustness under adversarial prompting. Seed prompts are first evaluated on the System Under Test (SUT) to obtain a reference grade, then transformed into adversarial variants and re-evaluated on the same SUT, and finally compared to quantify changes in performance and produce a safety-degradation report.

and because the benchmark supplies standardized prompts, shared evaluators, and calibrated grading rubrics. These properties enable comparison across systems and releases and support longitudinal analysis of safety degradation under adversarial prompting.

2.2 The MLCommons Jailbreak Benchmark v0.5

Building on the hazard-based evaluation framework provided by AILuminate, Jailbreak Benchmark v0.5 represents the first MLCommons effort to operationalize adversarial prompt testing against deployed LLMs at scale. Its primary objective is to quantify how safety-aligned systems degrade under adversarial pressure relative to a standardized baseline, thereby enabling comparative evaluation across models and supporting deployment review and risk-management use cases.

2.2.1 Scope

Version 0.5 restricts scope to single-turn, inference-time prompt attacks delivered through the standard user interface. The benchmark assesses safety in the domain of text perturbations to model inputs and measures safety performance by assessing text outputs. Safety performance is anchored to AILuminate v1.1, which supplies the hazard definitions and grading rubric used to establish baseline system behavior.

2.2.2 Operationalization of Testing

Operationally, v0.5 proceeds in three stages as shown in Fig. 1. Each system is first evaluated on a seed set of AILuminate prompts to obtain a reference safety grade. Mutated prompts representing known jailbreak techniques are then applied to the same systems. Finally, the benchmark computes the resulting degradation in safety scores using a shared evaluation pipeline and grading scale.

2.2.3 Observed Challenges

Experience with v0.5 informs the next stage of scaling jailbreak evaluation. While v0.5 provided broad coverage through descriptive groupings of adversarial prompts, the attack taxonomy presented here builds on that foundation by introducing a more explicitly mechanism-based organization, enabling more systematic coverage across attack strategies and clearer interpretation of results

in terms of underlying prompt-manipulation behaviors rather than surface features or outcome categories.

2.2.4 Lessons for Subsequent Releases

These observations motivate the methodological emphasis adopted in this work. In particular, v0.5 demonstrates that jailbreak evaluation benefits from an explicit, mechanism-level structure that governs which attacks are included, how they are sampled, and how results are reported.

2.3 Relationship to Prior and Future MLCommons Work

2.3.1 Continuity with Prior Benchmarks

The attack taxonomy presented in this paper builds directly on earlier MLCommons evaluations. From AILuminate v1.1, this work continues to rely on established hazard definitions, evaluator design, and grading philosophy [2]. From Jailbreak Benchmark v0.5, it retains the paradigm of measuring safety degradation under adversarial prompting, the reuse of shared evaluation pipelines, and a focus on single-turn inference-time attacks delivered via the prompt interface.

2.3.2 Staged Release Strategy

At the present stage, limitations in the evaluation protocol, particularly the absence of attack-stratified analysis of evaluator reliability, prevent the empirical results from supporting strong or stable claims about system robustness. Aggregate judging accuracy obscures systematic variation in evaluator behavior across heterogeneous prompt-bypass mechanisms, making it difficult to determine whether observed failures reflect general weaknesses in the judging pipeline or attack-specific blind spots. Under these conditions, releasing a comprehensive quantitative jailbreak benchmark or comparative performance results would risk misleading interpretation. Consequently, this stage of the work concentrates on formalizing a defensible taxonomy of single-turn prompt-bypass mechanisms and articulating a general methodology for taxonomy-guided jailbreak evaluation. The emphasis is placed on clarifying scope boundaries, classification rules, and design principles that are prerequisites for future reproducible and defensible security assessments, rather than on publishing leaderboards or finalized robustness metrics.

A subsequent Jailbreak Benchmark is intended to extend this foundation to comprehensive coverage across the taxonomy. In that release, benchmark results are expected to satisfy four criteria:

- reproducibility across organizations and evaluation environments;
- attack selection based on an explicit and defensible sampling strategy grounded in the taxonomy;
- generation of adversarial prompts using verified and auditable code;
- broad and systematic coverage of prompt-manipulation families;
- **evaluator reliability assessed at the level of individual attack families**, enabling detection of systematic judge failure modes and reducing bias introduced by heterogeneous prompt structures.

This staged approach mirrors the evolution of other MLCommons benchmarks, in which early releases establish methodological infrastructure and later versions expand scope while preserving comparability. By separating taxonomy construction from full coverage goals in v1.0, the roadmap aims to enable stable longitudinal evaluation and principled extension to additional modalities and interaction patterns in future releases.

3 Scope and Definitions

3.1 System and Threat Model

The methodology proposed in this work considers LLMs deployed in text-based, user-facing applications, consistent with the evaluation framing introduced in the MLCommons Jailbreak benchmark v0.5. Systems under test (SUTs) are modeled strictly as inference-time services: the analysis does not assume access to internal model weights, training data, safety-classifier implementations, system prompts, or deployment configurations beyond those exposed through the user-facing interface. The resulting threat model therefore adopts a black-box interaction setting commonly used in prior adversarial NLP and LLM robustness studies [25, 17, 26].

3.2 In-Scope Attack Categories

This release of the taxonomy includes only inference-time prompt attacks that can be executed through a single user turn and that operate by manipulating the content, structure, or representation of the submitted input. As in v0.5, the benchmark restricts attention to black-box interaction with deployed models through their standard prompt interfaces, without access to internal weights or training data. However, whereas v0.5 enumerates attack prompts primarily for coverage, we introduce a defensible and reproducible mechanism-based taxonomy.

The in-scope categories correspond to prompt-level bypass mechanisms that have been repeatedly observed in recent LLM security research and practitioner red-teaming exercises [27, 17, 26].

3.2.1 Direct jailbreak and policy-override prompts

This category includes explicit attempts to supersede system policies or safety constraints through override language, persona assignment, or multi-rule templates. Examples include commands that instruct the model to ignore prior instructions, adopt an unrestricted persona, or produce dual responses under alternative behavioral regimes. Such strategies form the basis of widely circulated jailbreak templates and have been documented across both academic studies and open benchmarking efforts [27, 26].

3.2.2 Prompt perturbations that preserve semantic intent

These attacks alter the surface form of a hazardous request while retaining its underlying meaning, with the goal of bypassing lexical or heuristic-based refusal triggers. The taxonomy includes both human-authored edits, such as paraphrasing, synonym substitution, and character-level obfuscation, and algorithmically generated perturbations optimized for transferability across models. Prior work on adversarial text generation and transferable jailbreak prompts motivates the inclusion of this class as a first-order benchmark dimension [25, 17, 28].

3.2.3 Encoding- and formatting-based prompt manipulations

This category covers attacks that hide or transform unsafe instructions through encoding, Unicode controls, structured containers, or rigid formatting schemes that alter how the model parses or interprets the input. Examples include Base64-wrapped directives, zero-width character insertion, ASCII disguises, JSON or Markdown wrappers, and positional prefix–suffix steering. Such techniques appear both in the adversarial NLP literature and in contemporary LLM red-teaming reports [17, 26], and therefore, warrant explicit representation in a benchmark-oriented taxonomy.

3.2.4 Composite and ordering-based prompts

These attacks rely on how instructions are arranged within a single input rather than on local textual obfuscation alone. They include benign contextual framing followed by a harmful core, scenario-based scaffolding, fragment recombination, interleaving of safe and unsafe segments, and algorithmic prompt-search procedures that iteratively refine payload structure. Because such strategies exploit global prompt composition rather than lexical properties, they form a distinct family that must be sampled separately in order to evaluate structural robustness [27, 17].

Across all in-scope categories, the proposed taxonomy requires that each concrete attack instance map to exactly one leaf category through explicit decision rules. Prompts that combine multiple bypass techniques in ways that prevent clear dominance of a single mechanism are not modeled in the present formulation and are instead reserved for future extensions of the taxonomy. This one–instance–to–one–leaf constraint enables deterministic labeling, reproducible corpus construction, mechanism-stratified analysis, and auditable evaluation design, thereby supporting longitudinal comparison across future jailbreak assessment efforts.

3.3 Out-of-Scope Attack Categories

This release deliberately restricts its coverage to single-turn, inference-time prompt attacks as an initial step toward building a systematic and reproducible evaluation framework. Several important classes of adversarial behavior, therefore, fall outside the scope of the present taxonomy and benchmark. These exclusions are explicit rather than incidental and are intended to clarify what the current evaluation results do and do not measure.

3.3.1 Multi-turn and conversational strategies

Attacks that rely on iterative manipulation across multiple interaction turns, such as gradual trust-building, staged escalation, or dialogue-based constraint erosion [26], are excluded from this release. Although such strategies represent an important real-world threat, their dependence on conversational state and adaptive attacker behavior complicates instance-level labeling and controlled benchmarking under the present methodology.

3.3.2 Training-time compromise

Attacks that target the training pipeline rather than inference-time behavior, including data poisoning, backdoored models, covert fine-tuning, and model-supply-chain compromise, remain outside scope. These techniques require different threat models, evaluation protocols, and access assumptions than those adopted here [29, 30, 21].

3.3.3 Infrastructure-level vulnerabilities

The taxonomy does not include attacks that primarily exploit weaknesses in surrounding systems rather than in prompt interpretation by the model itself. Examples include authentication failures, tenant isolation breaches, logging leakage, network misconfiguration, or privilege escalation within application infrastructure.

3.3.4 Indirect prompt injection through external channels

Attacks in which malicious instructions enter the model context through retrieval-augmented generation pipelines, tool outputs, uploaded documents, or external files are excluded from the present release. Such attacks involve cross-component interactions and provenance ambiguity that fall outside the prompt-only scope defined earlier [31].

3.3.5 Social-engineering–dominant attacks

Strategies that depend primarily on human deception or manipulation, rather than on properties of the prompt interpreted by the model, are not covered. Examples include phishing or persuasion campaigns whose success hinges on downstream user behavior rather than on bypassing model-level safety constraints [32].

By articulating these exclusions explicitly, the benchmark confines its claims to a well-defined threat surface and preserves comparability with prior releases while providing a clear boundary for future extensions.

3.4 Terminology and Conceptual Distinctions

To ensure consistency across taxonomy construction, benchmark execution, and result interpretation, this work adopts a set of precise technical definitions and conceptual separations drawn from prior MLCommons releases and contemporary safety and security standards. Where applicable, terminology follows the conventions introduced in the MLCommons Jailbreak Benchmark v0.5 and the AILuminate assessment standard, while extending them to support a mechanism-centered taxonomy.

- **Jailbreak:** We follow the definition articulated in MLCommons Jailbreak Benchmark v0.5, which in turn aligns with emerging ISO terminology: a *jailbreak* denotes a user-authored, inference-time prompt intended to circumvent safety constraints and induce otherwise restricted model behaviors. In this framing, jailbreaks are a special case of prompt injection, restricted to safety-policy evasion rather than broader task manipulation.
- **Hazard, Attack, and Risk:** Consistent with the AILuminate safety standard and OECD-aligned systems-engineering terminology, we distinguish three concepts that are frequently conflated in informal discussion:
  - **Hazard**: a potential source of harm defined by content class or outcome (e.g., violent wrongdoing, privacy violation).
  - **Attack**: a strategy or mechanism used to induce hazardous behavior, such as encoding abuse or persona-based overrides.

  - **Risk**: the combination of the likelihood that a hazardous outcome occurs and the severity of its potential consequences.

  This taxonomy classifies *attacks*, whereas the AILuminate benchmark organizes *hazards*. The two dimensions are deliberately orthogonal: a single attack mechanism may target multiple hazard classes, and the same hazard may be reachable through several attack families.

- **Safety Evaluation versus Jailbreak Evaluation:** Following the separation emphasized in MLCommons Jailbreak Benchmark v0.5, we draw a sharp distinction between two forms of assessment:
  - **Safety evaluation** measures whether a system produces policy-violating outputs when presented with ordinary or minimally adversarial prompts, as operationalized in AILuminate.
  - **Jailbreak evaluation** measures a system's robustness to adversarial attempts designed to induce such violations, reporting the degradation in safety performance under attack.

  The v0.5 release operationalized this distinction through paired baseline and adversarial measurements and the *resilience gap* metric. Version 0.7 retains this conceptual separation while introducing a defensible, mechanism-first taxonomy to stratify attacks more systematically.

4 Design Requirements for a Benchmark-Operational Taxonomy

A jailbreak benchmark designed to support reproducible security evaluation and standards-aligned governance must treat attack classification as a primary methodological decision rather than as an auxiliary descriptive exercise. The taxonomy governs which prompt-based attacks are instantiated, how representative corpora are constructed from an unbounded design space, and how evaluation results are partitioned for mechanism-level analysis. In this role, the taxonomy functions as part of the benchmark's experimental apparatus, not as an interpretive overlay applied after testing.

4.1 Benchmarking Constraints

Benchmark design imposes concrete technical constraints on any taxonomy of prompt-based attacks. In this setting, categories determine how corpora are constructed, how instances are labeled, how coverage is quantified, and how results are stratified for analysis [33]. These constraints operationalize the benchmark-credibility properties introduced earlier in Section 1: robustness, reproducibility, and defensibility, by translating them into concrete requirements on classification, sampling, and category evolution. A taxonomy that does not satisfy these requirements cannot support reproducible evaluation or defensible claims about system robustness.

The first constraint is the requirement for **deterministic instance labeling**. Each attack prompt must map to a single category through explicit decision rules, as emphasized in adversarial machine learning taxonomies that require unambiguous class membership for systematic evaluation. Independent annotators applying the taxonomy must reach the same classification outcome for the same input, since reproducibility and inter-annotator agreement are prerequisites for meaningful robustness analysis [34]. This property is necessary for computing stable per-category success rates

and for enabling comparison across systems and benchmark releases [35]. Overlapping categories, underspecified boundaries, or inconsistent abstraction levels introduce label variance that contaminates reported metrics and prevents meaningful mechanism-level interpretation, a failure mode repeatedly identified in the robustness evaluation literature [34, 35].

The second constraint is the requirement for **representative sampling across attack mechanisms**. Prior work in adversarial machine learning emphasizes that the space of possible attacks is combinatorial, with individual strategies admitting many surface realizations, making exhaustive enumeration infeasible in practice [21, 34]. Consequently, robustness benchmarks must rely on taxonomy-driven sampling procedures to approximate coverage of the adversarial space. Categories must therefore reflect the underlying mechanism by which a prompt bypasses safeguards rather than superficial stylistic patterns or widely publicized templates, consistent with recommendations in prompt security surveys and threat modeling frameworks [4, 18]. Oversampling familiar jailbreak styles biases robustness estimates, while undersampling less visible but structurally distinct mechanisms narrows the scope of conclusions that can be drawn about general system resilience, a risk highlighted in robustness evaluation studies and red teaming analyses [35, 21].

The third constraint is the requirement for **category stability over time**. Benchmarks in adversarial machine learning are intended to support longitudinal analysis across evolving models and defenses, which requires that category definitions remain semantically stable as new attacks emerge [21, 34]. The taxonomy must therefore permit the introduction of novel attack mechanisms without retroactively reassigning previously collected instances, a property emphasized in formal taxonomy-engineering methods that stress fixed internal semantics and controlled refinement through leaf-level extension [24, 36]. Internal nodes must retain stable meaning, while refinement proceeds through the addition of new leaves that capture genuinely new prompt-manipulation strategies rather than reorganizing existing classes [24]. Without this property, apparent performance trends confound genuine system improvement with shifts in classification structure, undermining the interpretability of robustness trajectories in evolving benchmarks [34, 21].

4.2 Why Existing Taxonomies Are Insufficient

Existing LLM security taxonomies provide valuable conceptual foundations for governance, architectural reasoning, and threat modeling. Nevertheless, when these frameworks guide the construction and evaluation of jailbreak benchmarks, structural mismatches emerge. Most prior taxonomies organize threats around impact categories, system components, or lifecycle stages rather than around the concrete prompt-level mechanisms that benchmarks must instantiate and measure.

- **Risk-centric taxonomies:** Risk-oriented frameworks classify attacks according to potential harm or misuse outcomes, such as data leakage or policy evasion, as in Weidinger et al. [18], OWASP [37], CSA [38], Cisco [39], FS-ISAC [40], and NIST [10]. This organization supports governance workflows, risk registers, audit reporting, and communication with non-technical stakeholders. However, outcome-based categories remain too coarse for instance-level benchmark construction. Distinct prompt-manipulation strategies often map to the same risk label, which prevents evaluators from distinguishing whether failures arise from encoding-based obfuscation, compositional instruction hijacking, or role-play prompting. Without mechanism-level resolution, such frameworks cannot guide corpus sampling or support reproducible placement of individual prompts into non-overlapping evaluation categories.

- **Lifecycle- and architecture-based frameworks:** System- and lifecycle-oriented taxonomies organize threats by application component or development phase, as in Cui et al. [41] and Wang et al. [42]. This perspective facilitates ownership assignment and control mapping across engineering teams. At the same time, it places prompt-level jailbreak attempts alongside training-time poisoning and infrastructure vulnerabilities that require different testing methodologies. This conflation prevents benchmarks from isolating prompt robustness as a measurable property and introduces ambiguity about which attack surface an evaluation result reflects.
- **Implications for corpus construction:** When taxonomies mix organizing principles or operate at inconsistent abstraction levels, corpus construction becomes ill-defined, a pattern documented across prompt-security surveys by Shayegani et al. [43], Yao et al. [44], Das et al. [45], and Li and Fung [46]. Overlapping categories allow the same prompt to fit multiple branches, which undermines deterministic labeling and inter-annotator agreement. High-level risk labels coexist with low-level technique descriptions, complicating sampling strategies and obstructing efforts to guarantee balanced coverage across bypass mechanisms. Ambiguous placement rules further prevent extension of the corpus without reclassifying historical instances, weakening longitudinal comparability across benchmark releases.
- **Implications for metrics and reporting:** Structural ambiguity at the taxonomy level propagates directly into evaluation metrics, as noted in benchmarking-oriented analyses by Hong et al. [5] and Sorokoletova et al. [12]. Overlapping or outcome-driven categories aggregate heterogeneous attack strategies, which renders per-category success rates difficult to interpret. Reorganization of branches across benchmark versions destabilizes longitudinal comparisons. Results also resist stratification by bypass mechanism, preventing diagnosis of which defenses fail and obstructing systematic analysis of progress across mitigation approaches.

These limitations motivate a mechanism-first taxonomy designed explicitly for benchmark operation rather than for governance or architectural reasoning. Prompt-focused studies by Derner et al. [47], Rossi et al. [48], Rababah et al. [4], Hong et al. [5], and Schulhoff et al. [26] demonstrate partial movement in this direction, but no existing framework enforces prompt-only scope, deterministic placement, stable splitting rules, and family-level coverage simultaneously.

4.3 Taxonomy Requirements

The benchmarking constraints identified in Section 4.1 arise from a persistent mismatch between the objectives that motivate most existing LLM attack taxonomies and the methodological requirements of instance-level jailbreak evaluation as discussed in 4.2. Prior classification efforts fall into several broad families. Risk-centric and standards-driven frameworks emphasize governance, auditability, and control alignment, as in [18, 37, 38, 10, 39, 40]. System- and module-oriented perspectives organize threats by application-layer boundaries in order to support mitigation ownership [41]. Lifecycle- or scenario-based surveys classify attacks by development stage so that assurance activities integrate into engineering processes [42, 49].

Alongside these governance- and architecture-focused approaches, a substantial body of literature has developed prompt-focused taxonomies that aim to support empirical testing and red-teaming. These include interaction-pipeline models proposed by Derner et al. [47], prompt-injection cate-

Table 1: Evaluation-Driven Requirements for a Prompt-Attack Taxonomy Used in Jailbreak Benchmarking

Requirement	Design Principle	Benchmarking Problem Addressed	Representative Failure Mode Without the Requirement
R1. Prompt-only scope	Restrict evaluation to single-turn inference-time prompt manipulation rather than training-time poisoning, infrastructure compromise, or multi-turn social engineering.	Prevents conflation of prompt robustness with upstream system design decisions, enabling clean attribution of failures and interpretable measurements.	A benchmark mixes jailbreak prompts with fine-tuning data poisoning and tool-interface exploits, making it impossible to determine whether failures stem from prompts or system configuration.
R2. Mechanism-first top level	Organise categories by bypass mechanism rather than harm type or architectural component.	Aligns taxonomy with mitigation design and enables mechanism-stratified testing and analysis.	Attacks group by outcome class (e.g., "policy evasion"), collapsing encoding- and perturbation-based bypasses and obscuring which defences fail.
R3. One-instance → one-leaf mapping	Require each prompt to map to exactly one leaf category.	Ensures unambiguous labelling, reproducible statistics, and high inter-annotator agreement.	A prompt appears in multiple categories, inflating coverage metrics and breaking cross-release comparability.
R4. Consistent splitting rule per node	Apply a single criterion at each internal node when subdividing categories.	Preserves mutual exclusivity, extensibility, and longitudinal stability.	Different levels split by intent, format, and strategy, producing ambiguous placements for new attacks.
R5. Reproducible, executable inclusion	Ground each category in concrete, runnable prompt instances.	Supports empirical benchmarking, auditability, and independent verification.	Categories remain abstract (e.g., "indirect reasoning abuse") and cannot be operationalised into benchmark suites.
R6. Coverage of major bypass families	Span dominant classes of prompt-manipulation strategies, including encoding, translation, and compositional attacks.	Avoids overfitting to narrow jailbreak styles and ensures representative robustness estimates.	The benchmark focuses on role-play jailbreaks and omits attack families responsible for many real-world failures.

gorizations by Rossi et al. [48], systematizations of prompt hacking and jailbreak strategies by Rababah et al. [4] and Sorokoletova et al. [12], multi-level surveys of prompt security by Hong et al. [5], and competition-driven corpora introduced by Schulhoff et al. [26]. Broader surveys synthesize these perspectives while spanning additional attack surfaces, including Chowdhury et al. [50], Esmradi et al. [51], Shayegani et al. [43], Yao et al. [44], Das et al. [45], and Li and Fung [46]. Domain- or goal-specific taxonomies further illustrate how classification structure responds to downstream requirements, as in the CIA-based taxonomy of Jones et al. [52] and the education-focused framework of Zahid et al. [53].

These bodies of work pursue distinct operational purposes, which in turn drive their structural choices. Governance-oriented frameworks privilege broad, communicable abstractions that populate risk registers and control catalogs [18, 10]. Architectural perspectives emphasize system boundaries so that mitigations map cleanly onto engineering teams [41]. Lifecycle-based approaches emphasize development-stage alignment so that security testing integrates into release processes [49]. Prompt-focused taxonomies emphasize technique-level resolution in order to support reproducible experimentation, red-team exercises, and corpus construction [47, 5].

Because the present benchmark focuses on single-turn, prompt-only attacks and relies on taxonomy-guided corpus construction, it adopts a prompt-focused perspective and distills a minimal set of criteria governing how categories function within an evaluation pipeline. These criteria do not attempt to unify all taxonomy traditions. Instead, they isolate the properties required for deterministic instance labeling, stable category growth across releases, and representative mechanism-level sampling.

The six requirements R1–R6 group into three evaluation-critical dimensions. R1 and R2 specify the scope and organizing principle. R3 and R4 enforce labeling determinism and structural stability. R5 and R6 govern operationalization into executable corpora and coverage across bypass families.

These six requirements derive jointly from recurring limitations observed across prior taxonomies, including scope mixing, overlapping category structures, and insufficient guidance for empirical labeling. This fact is summarized by Shayegani et al. [43], Yao et al. [44], Das et al. [45], and Li and Fung [46], and from the operational constraints of industrial-scale benchmarking, which demand unique instance placement, reproducible execution, and balanced mechanism-level coverage.

Table 1 enumerates each requirement, the benchmarking failure mode it prevents, and representative scenarios that arise when the requirement is absent. Further, Table 2 maps major families of prior frameworks to R1–R6 and demonstrates that most existing taxonomies satisfy only subsets of the criteria required for instance-level evaluation. Several governance- and risk-oriented LLM security frameworks (e.g., standards bodies and industry threat models) are omitted from Table 2 because they do not satisfy the prompt-only, mechanism-level, and instance-operational requirements needed for jailbreak benchmarking [18, 41, 42, 39, 37, 38, 10, 40]. These frameworks remain valuable for audit, governance, and system-level risk analysis, but their level of abstraction and scope make them unsuitable for corpus construction or mechanism-stratified evaluation. A taxonomy engineered for corpus construction and evaluation, therefore, remains necessary to support reproducible jailbreak benchmarking. The final row of Table 2 characterizes the taxonomy introduced in this benchmark. Unlike governance-oriented or architectural threat models, the present taxonomy satisfies all six requirements simultaneously.

5 Taxonomy of LLM's Single Turn Jailbreaks

Unlike prior releases that center on end-to-end grading pipelines, we organize evaluation around a mechanism-level classification of jailbreak techniques intended to serve as a stable foundation for prompt corpus construction, controlled evaluation, and future scoring pipelines.

Across all categories, the taxonomy is unified by a single guiding meta-characteristic: the ways in which an attacker modifies a single input prompt at inference time to induce a system to violate its intended safety or policy constraints. This framing excludes multi-turn social engineering, retrieval-augmented injection, and infrastructure-level compromises, and instead focuses on prompt-level manipulations that can be instantiated, audited, and evaluated in isolation. The taxonomy is further engineered to satisfy the benchmark-operational requirements defined in Section 4. In particular, it enforces deterministic placement rules for attack instances, supports one-to-one mappings between concrete prompts and leaf categories, and enables balanced sampling across heterogeneous attack families. These properties support reproducible dataset releases, interpretable reporting across mechanisms, and longitudinal comparisons as the benchmark evolves. Finally, each leaf category

Table 2: Evaluation-driven taxonomy requirements for prompt-only, instance-level jailbreak benchmarking and the extent to which major families of prior LLM security frameworks satisfy each requirement. A checkmark indicates that a framework family commonly supports the corresponding criterion.

Paper / framework	R1 Prompt-only scope	R2 Mechanism-first top level	R3 One-instance → one-leaf	R4 Consistent splitting rule per node	R5 Reproducible/ executable inclusion	R6 Coverage of major prompt-bypass families
Derner et al. [47]	✓					
Rossi et al. [48]	✓					
Rababah et al. [4]	✓					✓
Hong et al. [5]					✓	✓
Sorokoletova et al. [12]	✓	✓			✓	
HackAPrompt (Schulhoff et al.) [26]	✓				✓	
Peláez-González et al. [54]	✓					
Jones et al. [52]	✓					
MLC Attack Taxonomy	✓	✓	✓	✓	✓	✓

is defined in terms of executable prompt transformations rather than abstract threat descriptions. This design allows the taxonomy to integrate directly into prompt generators, corpus construction workflows, and evaluation pipelines in subsequent releases. In this sense, we establish the structural backbone of the jailbreak benchmark: a controlled vocabulary of mechanisms on which future measurement, grading, and governance artifacts will be built.

5.1 Methodological Process for Taxonomy Development

The process adapts established taxonomy-construction methods from information systems research to the operational constraints of jailbreak benchmarking introduced in Section 4, including deterministic instance labeling, representative mechanism-level coverage, stable category evolution, and benchmark reproducibility (R1–R6). Consistent with prior methodology work by Nickerson et al. [24], Usman et al. [55] Omair and Alturki [56], and Oberländer et al. [57], the workflow separates general design instructions from their concrete operationalization in this release.

The development process proceeds through eleven stages, each of which contributes to constructing a mechanism-first, instance-level taxonomy suitable for corpus construction and longitudinal evaluation.

5.1.1 Plan the Domain, Purpose, and Shape of the Taxonomy

The taxonomy-development process begins by fixing the fundamental design parameters that determine every subsequent modeling decision. Because a benchmark-operational taxonomy must support deterministic labeling, balanced corpus construction, and stable longitudinal evaluation, the first step translates these evaluation-driven constraints into explicit boundary conditions for classification [24, 34]. Without this grounding, later structural refinements risk drifting toward

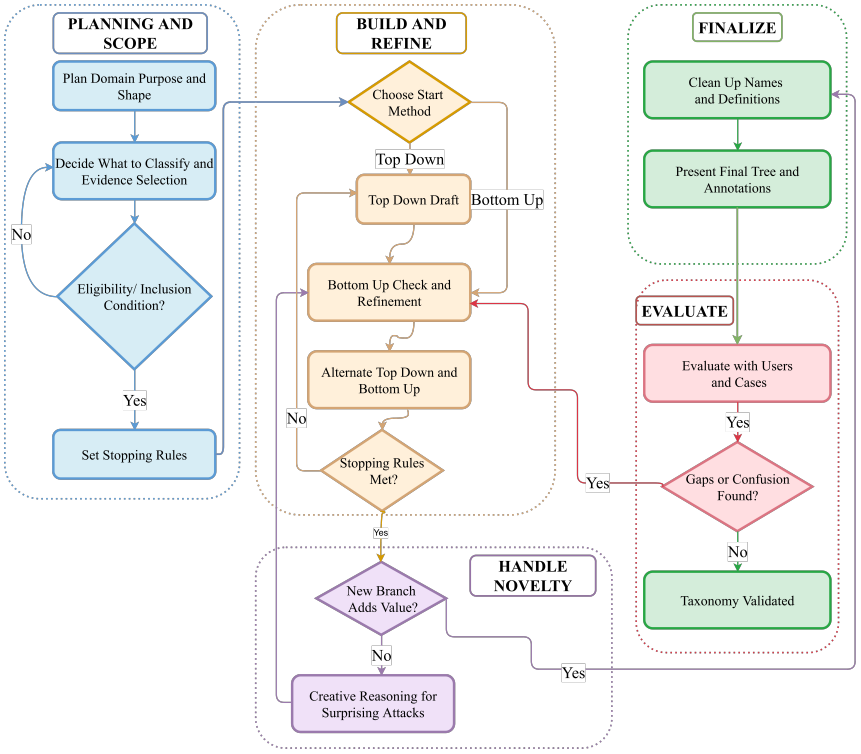

Figure 2: End-to-end methodology for constructing and validating the prompt-bypass taxonomy.

outcome-based groupings or mixed abstraction levels that undermine reproducibility and comparability in robustness measurement [24, 21].

At this stage, the process addresses four coupled design questions: which portion of the LLM attack surface the taxonomy covers; which communities rely on it and for what downstream tasks; which meta-characteristic governs all future categorical distinctions; and whether the resulting classification scheme adopts a single hierarchical structure or multiple orthogonal dimensions [24, 36]. Resolving these questions at the outset ensures that evidence collection and iterative refinement proceed against a stable target.

To ensure that these foundational design questions receive systematic and replicable treatment, the methodology first specifies a set of general design instructions that any benchmark-oriented taxonomy-development effort must satisfy [24, 36].

Instruction:

- Define the taxonomy's scope by stating explicitly what is included and excluded, including non-goals.
- Identify the intended users and the decisions or tasks the taxonomy is designed to support.
- Specify the meta-characteristic that governs all subsequent categorical distinctions.
- Determine the structural form of the taxonomy (e.g., single hierarchy versus multiple independent dimensions).

Operationalization in this work: Having fixed these design commitments, the present release instantiates them as follows:

- **Domain and scope:** The taxonomy is restricted to attacks that intentionally misuse or exploit a deployed LLM at inference time through a single prompt or small perturbations to that prompt in order to elicit unintended or policy-violating responses. Included attacks encompass direct jailbreak prompts and policy-override templates, as well as adversarial text transformations that preserve semantic intent while altering surface form.

  Several categories remain explicitly out of scope in this release and are reserved for future extensions. These include training-time attacks and fine-tuning manipulation (such as data poisoning and backdoored models), multi-turn conversational social-engineering strategies, tool-poisoning attacks, infrastructure exploits whose success derives primarily from system misconfiguration rather than prompt manipulation, and indirect prompt attacks mediated through retrieval-augmented generation pipelines.

- **Intended users and purpose:** Within these boundaries, the taxonomy targets LLM security teams, red-team practitioners, product engineers, and academic researchers. It supports three benchmark-critical tasks: describing single-turn prompt attacks using a shared and precise vocabulary; comparing distinct prompt-manipulation strategies at the mechanism level; and designing, prioritising, and interpreting defensive evaluations for jailbreak robustness.

- **Meta-characteristic:** Following Nickerson et al. [24], the taxonomy adopts a single governing principle that all branches must respect: *the manner in which an attacker manipulates or perturbs a single prompt (or prompt-adjacent input) at inference time to cause an LLM to violate its intended safety or security constraints for that interaction.* Every internal node therefore corresponds to a difference in prompt-manipulation mechanism rather than to attacker intent, harm category, or system component.

- **Shape of the taxonomy:** On the basis of these commitments, the development process selects a mechanism-first hierarchical structure rather than multiple independent, non-hierarchical dimensions. A single dominant axis of classification simplifies instance placement, enables deterministic leaf assignment, and supports corpus construction and mechanism-stratified reporting, as required for benchmark operation.

5.1.2 Decide What to Classify and How Evidence Is Selected

Once the domain boundaries and organizing principle are fixed, the taxonomy-development process turns to the question of what constitutes a classificatory object and how the empirical evidence for such objects is assembled. Because the taxonomy aims to support benchmark construction rather than high-level risk communication, this stage focuses on identifying reusable classes of single-turn

prompt attacks and on selecting literature and practitioner sources through a transparent and systematic procedure. These decisions directly influence coverage, representativeness across bypass mechanisms, and the reproducibility of future corpus releases.

This step, therefore, formalizes two tightly coupled design choices. First, it specifies what constitutes an attack instance for classification, ensuring that leaves correspond to stable manipulation patterns rather than idiosyncratic prompts. Second, it defines a PRISMA-style selection protocol that governs how candidate attacks are identified, screened, and admitted into the evidence base that drives structural refinement.

Before describing how this process is instantiated in the current release, the methodology articulates the general instructions that govern object definition and evidence selection in any benchmark-oriented taxonomy development effort.

Instruction:

- Define each classified object as a distinct class of single-turn prompt attack, such as a jailbreak template that consistently bypasses safeguards, a recurring pattern of adversarial perturbations or encoding tricks, or a specific way of inducing unsafe action execution within one turn.
- Adopt a PRISMA-style literature-selection protocol to avoid biased sampling, including: defining databases and search strings in advance; running searches and collecting records (identification); removing duplicates and screening titles and abstracts using explicit inclusion and exclusion criteria (screening); reading remaining papers in full to verify that they describe single-turn, inference-time, prompt-based or perturbation-based attacks (eligibility); and retaining only those sources that match the scope and provide sufficient detail to identify a distinct manipulation type (inclusion).
- Ensure that this process yields a transparent and systematic evidence base rather than an ad hoc collection of examples.

Operationalization in this work: Having fixed these evidence-selection commitments, the present release instantiates them as follows:

- **Objects (instances):** Each classified object corresponds to a reusable class of single-turn prompt attack rather than to a single handcrafted prompt. Examples include jailbreak templates that consistently override refusal behavior, recurring perturbation strategies such as adversarial typos or symbol insertions, Unicode- or tokenization-based encoding tricks that induce policy evasion, and prompting patterns that trigger unsafe tool or action execution within a single inference call.
- **Sources and time window:** We query Google Scholar, IEEE Xplore, ACM Digital Library, and arXiv, and review curated GitHub repositories and online lists of LLM attacks, including collections focused on jailbreaks, prompt injection, adversarial text, poisoning, covert fine-tuning, and training-data extraction. Curated lists are included because several high-impact jailbreak and injection techniques appear in practitioner channels during 2023–2025, before indexed publication.

  The search window spans 2015–2025. Because LLM security research often disseminates preprints or technical reports before formal venue publication, eligibility is determined by

the first public availability date, with later venue publication dates recorded separately to preserve chronological fidelity.

- **Search strategy:** Search queries combine three term families: (i) model or target terms (e.g., "LLM," "foundation model," "GPT"); (ii) attack or vulnerability terms (e.g., "jailbreak," "prompt injection," "adversarial text," "Unicode smuggling," "data poisoning," "backdoor," "prompt stealing," "PII extraction"); and (iii) scope terms (e.g., "safety," "alignment," "robustness," "security," "evaluation"). A representative query is:

  ("large language model" OR LLM OR GPT) AND (adversarial OR "prompt injection" OR jailbreak OR "data poisoning") AND (safety OR alignment OR security).

- **Inclusion criteria:** A source enters the corpus only if it (a) falls within the defined time window and is written in English, (b) defines at least one LLM attack pathway that either operates directly at inference time or can be expressed as prompt manipulation under the present scope, and (c) provides sufficient methodological detail to support mapping into category, subcategory, and prompt-only sub-subcategory. Red-team reports and vendor case studies focused on inference-time misuse are included where available.

- **Quality signals:** Priority is given to instances that articulate a clear threat model, evaluate aligned or safety-tuned models, release reproducible artifacts, and/or present quantitative evidence of bypass success or impact.

- **Exclusion criteria:** Studies are excluded when attacks do not transfer to modern LLMs, involve benign noise without adversarial intent, focus exclusively on defensive techniques without reusable attack patterns, lack detail for unambiguous placement, or require multi-turn execution.

- **PRISMA reporting:** The resulting identification, screening, eligibility, and inclusion process is summarized using a PRISMA flow diagram, presented in Fig. 3.

- **High-profile attack governance:** In addition to the PRISMA-based literature set, attacks that rapidly become prominent within the LLM security community are admitted under a lightweight governance rule. An attack is triggered when the security workstream records consensus that at least one of the following signals is present: substantial coverage by reputable press or incident reporting; rapid uptake in practitioner channels such as vendor advisories, red-team disclosures, or major open-source repositories; or explicit benchmarking or standardization interest from external bodies. These sources are documented separately so that the core literature sample remains clearly defined.

- **Sampling strategy:** Within the included corpus, sampling proceeds primarily by prompt-manipulation mechanism. Coverage targets attacks that rely on overt natural-language carriers (e.g., role-play jailbreaks), perturbation-based manipulations, encoding-based tricks, and combinations of these strategies within a single prompt. Additional examples are added in later rounds to close coverage gaps and stress-test emerging branches.

5.1.3 Set Stopping Rules

After defining the classification and assembling the empirical evidence base, the taxonomy-development process requires explicit criteria for determining when refinement cycles terminate and the struc-

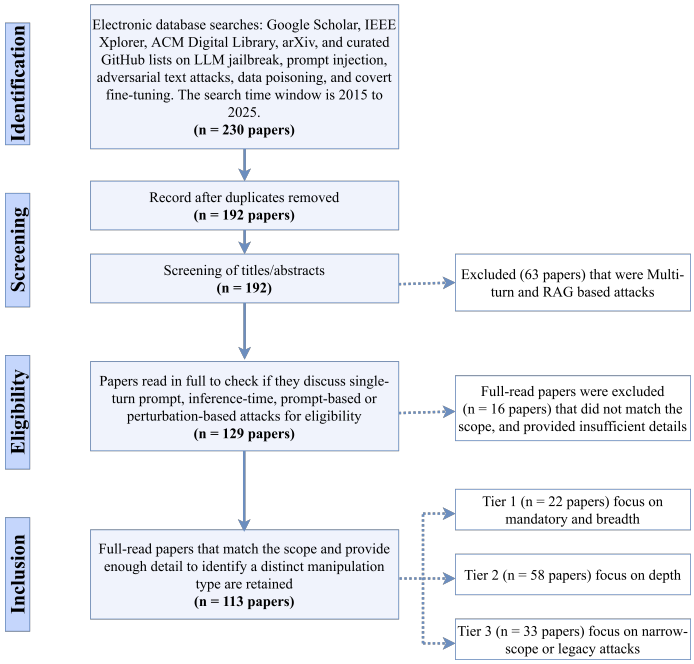

Figure 3: PRISMA-style literature review and paper selection pipeline used to construct the taxonomy and Tier-1 attack set. The flowchart shows database search, duplicate removal, screening, eligibility assessment, and final inclusion, together with exclusion criteria and the tiered classification of retained papers.

ture stabilizes. Because the taxonomy must support reproducible benchmarking and longitudinal comparison across releases, convergence cannot rely on informal judgment alone. Therefore, there is a need to formalize both objective and subjective stopping rules that govern when the hierarchy is treated as sufficiently complete, non-overlapping, and operational for corpus construction.

The stopping rules play two roles in the overall pipeline. First, they prevent uncontrolled growth of the taxonomy through continual subdivision in response to marginal cases. Second, they ensure that every retained category remains empirically grounded in real attacks and that every collected instance admits a unique placement, thereby enforcing requirements R3–R6.

Before describing the concrete convergence criteria applied in this work, the methodology specifies the general stopping-rule instructions that any benchmark-oriented taxonomy-development effort must follow.

Instruction:

- Define objective stopping rules that trigger termination of refinement when no new node type is required, no node needs to be split or merged, no two nodes represent the same prompt-manipulation mechanism, every leaf and major internal node is illustrated by at least one real attack example drawn from the literature or documented cases, and every collected attack maps to exactly one leaf of the tree.
- In addition, define subjective stopping rules that require the taxonomy to remain practically usable, with bounded depth and breadth, distinctions that are meaningful to practitioners, and coverage that is sufficient for contemporary prompt- and perturbation-based attacks. Treat violations of these criteria as triggers for further refinement.

Operationalization in this work: Having fixed these convergence commitments, the present release instantiates them as follows:

- **Objective stopping rules:** After each full refinement cycle, we re-project the complete instance set and check whether (i) no new top-level branch or node type is required; (ii) no additional split or merge is necessary to maintain clear category boundaries; (iii) no two nodes encode the same prompt-manipulation mechanism; (iv) every major internal node and every leaf is supported by at least one concrete attack instance drawn from the PRISMA-derived corpus or other documented cases; and (v) every collected attack maps to exactly one leaf in the hierarchy.
- **Subjective stopping rules:** We also require that the resulting tree remains manageable in size and depth, avoiding both excessively deep and narrow structures and overly flat hierarchies with dozens of leaves. Branch distinctions must remain intelligible to practitioners engaged in red-teaming and benchmark design. Finally, the taxonomy is treated as sufficiently complete for the current state of knowledge when all corpus attacks admit non-overlapping placement at the leaves, unless multi-label assignment is explicitly designed for a particular branch. Any violation of these conditions triggers further revision of the taxonomy.
- **Stability checks after refinement cycles:** The taxonomy is declared stable only when repeated application of the above rules yields no additional structural changes. In particular, stability requires that no new instance necessitates the introduction of a new top-level branch, no further split or merge is required to preserve clarity, every internal node includes an explicit splitting rule or axis sentence, every leaf includes a clear definition, and every attack instance maps to exactly one leaf without resorting to catch-all categories.

5.1.4 Select a Construction Strategy and Draft an Initial Hierarchy

With the evidence base assembled and stopping rules defined, the taxonomy-development process next determines how conceptual structure and empirical instances interact during early construction. Taxonomy-methodology research emphasizes that classification schemes may originate either from theoretical reasoning or from data-driven clustering, but benchmark-oriented settings impose additional constraints: the initial structure must admit deterministic instance placement, avoid topic-level groupings, and remain amenable to systematic refinement through repeated projection of real attacks.

This step, therefore, governs how the first candidate hierarchy is produced and how it is stress-tested

against empirical instances. The choice between top-down, bottom-up, or hybrid construction directly affects the interpretability of early branches, the rate at which coverage gaps emerge, and the likelihood that the final structure converges to a stable, mechanism-first organization compatible with requirements R2–R4.

Before describing the concrete approach adopted in this work, the methodology outlines the general construction strategy instructions derived from prior taxonomy research.

Instruction: Decide whether to begin by drafting a conceptual hierarchy (top-down), clustering real attack instances first (bottom-up), or combining both approaches. If combining, draft an initial tree and immediately test it against real examples, using placement failures to drive revisions. Following Nickerson et al., allow either top-down proposal of an initial tree from theory and experience or bottom-up growth of a tree from clustered empirical attacks.

Operationalization in this work: Having fixed these construction-strategy commitments, the present release instantiates them as follows.

We commence with a simple top-down draft informed by existing prompt-security taxonomies and practitioner knowledge, then refine it through systematic bottom-up testing against the collected corpus. Rather than clustering by surface topic or wording, instances are grouped through a mechanism-first lens: each prompt is assigned to the primary bypass mechanism by which unsafe intent passes model guardrails.

From these clusters, the development process constructs a single hierarchical tree by enforcing a consistent splitting rule at every internal node. For each mechanism class, the first split follows the most informative "how-it-works" axis, such as surface-form perturbation versus representation-level encoding transformation, explicit override rhetoric, or compositional assembly of prompt fragments. Sub-branches then refine these classes into mutually exclusive subtypes based on construction strategy and granularity, for example, character-level versus paraphrase-level perturbations, Unicode or tokenization tricks versus schema wrappers, and benign framing versus fragment recombination or optimization-based assembly.

This procedure yields four top-level branches, Perturbation, Encoding Abuse, Overt Carriers, and Composition & Ordering, each accompanied by an explicit axis sentence that justifies the split and preserves unique placement and extensibility under the stopping rules defined in Step 3.

5.1.5 Top-Down Draft of the Prompt-Manipulation Tree

Having fixed the construction strategy and convergence criteria in the previous step, the taxonomy-development process next produces a concrete top-down draft of the prompt-manipulation hierarchy. This stage converts the previously defined meta-characteristic into an explicit tree structure whose nodes admit deterministic placement and principled refinement under the established stopping rules. The emphasis here is not on discovering new attack families but on expressing existing knowledge in a way that exposes clear boundaries, consistent splitting rules, and extensible internal structure.

This step focuses on specifying the root of the tree, identifying a small number of first-level partitions, and recursively expanding lower levels while enforcing mutual exclusivity and explicit differentiation criteria at every internal node.

Before describing the concrete instantiation in this release, the methodology articulates the general top-down drafting instructions that govern this stage of taxonomy construction.

Instruction: Define the root node directly from the scope statement. Identify a small set of first-level categories that partition the space in a manner consistent with the governing meta-characteristic. Expand the hierarchy using consistent splitting rules, ensuring that children are mutually exclusive and collectively meaningful. Ensure that every internal node specifies an explicit criterion that explains how its children are differentiated.

Operationalization in this work: Having fixed these drafting commitments, the present release instantiates them as follows:

- **Root node:** The root represents single-turn, inference-time prompt attacks under the evaluation constraints established earlier in this section.
- **Level 1 categories:** Level 1 instantiates four prompt-manipulation families, *Perturbation*, *Encoding Abuse*, *Overt Carriers*, and *Composition & Ordering*, derived from existing knowledge and consistent with the meta-characteristic.
- **Lower-level refinement:** Each Level 1 branch is recursively refined into more specific prompt-manipulation patterns by applying one splitting rule per internal node, expressed as an axis sentence. These rules ensure that siblings are mutually exclusive and that membership depends solely on the observable properties of the prompt.

  Subcategories and, where applicable, prompt-only sub-subcategories capture stable construction variants within each family. Examples include character- versus paraphrase-level perturbations, Unicode or encoding-layer techniques versus structured wrappers, explicit override frames versus contextual deception, and concatenation or segmentation strategies versus ordering effects.

5.1.6 Execute Bottom-Up Refinement Cycles

Once a top-down draft tree is in place, the taxonomy-development process subjects it to systematic bottom-up testing against real attack instances. This stage operationalizes the empirical grounding principle articulated in Step 2 and ensures that the abstract hierarchy withstands contact with observed prompt-bypass techniques. By repeatedly projecting the collected corpus onto the draft tree, the process identifies structural weaknesses, overly broad categories, missing branches, and ambiguities that would otherwise undermine deterministic labeling.

This step therefore governs how instance placement drives structural revision and how successive refinement cycles converge toward a stable, mechanism-first taxonomy under the stopping rules defined in Step 3.

Before describing the concrete instantiation adopted in this release, the methodology articulates the general bottom-up refinement instructions that guide this stage of taxonomy construction.

Instruction:

- Execute a bottom-up cycle by classifying real single-turn prompt attacks from the literature and documented sources into the draft tree, placing each instance in the most suitable leaf. Flag attacks that fit more than one branch, fit no branch, or render a branch overly broad or vague.
- Look for natural groupings and tensions revealed by the data, such as cases where a category conflates distinct manipulation strategies and requires a split.
- Adjust the tree by splitting nodes that mix different manipulation types, merging nodes that cannot be reliably distinguished in practice, introducing new branches when examples reveal previously unmodeled patterns, and pruning branches unsupported by real or reasonably expected attacks.
- After each change, re-test instance placement, verify that every attack fits exactly one leaf, and check that sibling nodes remain clearly distinct. Ensure that this bottom-up stage provides empirical grounding consistent with prior guidance on taxonomy development.

Operationalization in this work: Having fixed these refinement commitments, the present release instantiates them as follows:

We map each extracted attack instance from Step 2 to exactly one leaf of the mechanism-based hierarchy, assigning placement based on the dominant prompt-manipulation strategy rather than on topical content or outcome. During placement, instances are flagged when they admit multiple plausible leaves, fail to match any existing branch, or cause a node to function as an overly broad catch-all.

Resolution proceeds by tightening definitions and boundary rules before performing structural edits. Only when definitional refinement proves insufficient do we introduce splits or merges, each justified by an explicit axis sentence. After each revision, all instances are reprojected onto the updated tree to verify their unique and stable placement.

This iterative process continues until the entire corpus admits deterministic classification, yielding an updated hierarchy, revised codebooks, and a change log documenting the taxonomy's evolution across refinement cycles.

5.1.7 Alternate Top-Down and Bottom-Up Refinement until the Tree Stabilizes

After executing initial bottom-up refinement cycles, the taxonomy-development process alternates between conceptual reassessment and empirical retesting to drive the hierarchy toward a stable fixed point. This stage ensures that the tree remains complete with respect to known prompt-manipulation strategies, avoids redundant or overlapping branches, and preserves interpretability as new examples are added to the corpus. Alternating between theoretical reasoning and data-driven validation also guards against prematurely freezing a structure that reflects only the early evidence set.

This step governs how repeated top-down questioning and bottom-up reclassification interact to close coverage gaps, simplify the hierarchy, and enforce the stopping rules defined earlier in this section.

Before describing the concrete stabilization procedure adopted in this work, the methodology articulates the general alternation instructions that guide this stage of taxonomy development.

Instruction:

- Conduct repeated cycles of top-down reasoning and bottom-up checking. Use security knowledge to ask whether obvious prompt-manipulation strategies are missing and to explore simple variation dimensions, such as whether the manipulation is visible or hidden and whether it operates in natural language or through formatting or encoding.
- Re-classify the growing set of attack instances from the PRISMA-derived corpus and other sources, and test whether newly introduced branches are populated and useful.
- At each cycle, remove overlaps where two leaves encode effectively the same type, fill gaps revealed by new examples, and keep the tree as small and clear as possible.
- Cease iteration only when both the objective and subjective stopping rules defined earlier are satisfied, including the absence of significant changes in the most recent round, unique placement of all examples, and a hierarchy that remains simple and meaningful.

Operationalization in this work: Having fixed these stabilization commitments, the present release instantiates them as follows.

At each iteration, we re-evaluate whether every instance maps to the correct top-level manipulation family, *Perturbation*, *Encoding Abuse*, *Overt Carriers*, or *Composition & Ordering*, and whether any branch conflates distinct prompt-manipulation patterns. When mixing occurs, we either sharpen the applicable boundary rule or introduce an additional split along the branch's governing axis, such as surface-form perturbation versus representation-level encoding change, overt carrier framing, or compositional ordering effects.

In parallel, we re-project the full corpus onto the updated tree, remove redundant leaves, and verify that newly introduced branches are populated by real attacks. These cycles repeat until the stopping criteria are satisfied and no further material structural changes occur.

5.1.8 Use Creative Reasoning for Surprising Prompt Attacks

Even after repeated top-down and bottom-up refinement cycles, some newly observed attacks may resist clean placement within the existing hierarchy. In such cases, minor definitional sharpening or local restructuring may prove insufficient to preserve unique placement and mechanism-first organization. Taxonomy-methodology research recommends a distinct stage of creative reasoning to accommodate genuinely novel manipulation strategies without destabilizing the overall structure.

This step governs how the taxonomy responds to out-of-distribution prompt attacks that expose previously unmodeled dimensions of manipulation, while preserving fidelity to the governing meta-characteristic and the benchmark-operational requirements established earlier in this section.

Before describing the concrete procedure adopted in this work, the methodology articulates the general creative-reasoning instruction that applies when conventional refinement fails.

Instruction:

- When a newly identified object does not align cleanly with any existing branch and minor definitional adjustments prove insufficient, apply creative reasoning to propose structural changes.
- Following Omair and Alturki [56], articulate what is genuinely new about the attack, suggest candidate new branches or splits, search for additional examples that share the same underlying mechanism, and insert and test the proposed branch within the tree.
- Retain the new structure only if it adds explanatory power and remains consistent with the governing meta-characteristic; otherwise, merge or remove it.

Operationalization in this work: Having fixed these creative-reasoning commitments, the present release instantiates them as follows:

- **Characterizing novelty:** For attacks that fail to fit existing leaves, we first articulate precisely what distinguishes the manipulation from known families, for example, whether the prompt exploits context-length truncation or instruction displacement rather than altering visible surface form.
- **Proposing new structure:** On the basis of this characterization, we formulate candidate new branches or splits, such as a hypothetical root-level extension capturing prompt-structure or context-limit manipulation.
- **Searching for corroborating cases:** We then examine the PRISMA-derived corpus and additional sources for other attacks that rely on the same underlying trick, thereby testing whether the candidate branch reflects a recurring mechanism rather than an isolated instance.
- **Insertion and evaluation:** Proposed branches are temporarily inserted into the hierarchy, the corpus is reclassified, and the effect on clarity, exclusivity, and tree complexity is evaluated.

A new branch remains only when it demonstrably increases explanatory power and preserves alignment with the meta-characteristic; that is, it continues to classify attacks according to how the prompt is manipulated. Otherwise, the structure is removed or merged into an existing branch.

5.1.9 Clean Up Names and Definitions

Once the hierarchy stabilizes under repeated refinement cycles, the taxonomy-development process turns to linguistic normalization and definitional precision. Because the taxonomy is intended for instance-level labeling by multiple organizations and annotators, inconsistent terminology, overlapping labels, or vague category descriptions would directly undermine reproducibility and inter-annotator agreement. This stage enforces terminological discipline and explicit boundary rules across the entire tree.

This step governs how node labels are standardized, how synonymous or ambiguously overlapping terms are reconciled, how concise definitions are produced for each category, and how neighboring branches are differentiated at the observable prompt level.

Before describing the concrete procedure adopted in this work, the methodology articulates the general language-cleanup instructions that apply once structural convergence has been reached.

Instruction:

- After the tree stabilizes, list all node labels and remove or merge synonyms, deciding how related terms such as "jailbreak," "prompt injection," and "policy override" map onto the hierarchy.
- Write short, plain-language definitions for every node that specify the covered prompt-manipulation mechanism and include an inclusion example.
- Clarify boundaries between neighboring nodes by stating the key differentiating cue, such as "direct override in user prompt" versus "instruction hidden in retrieved document."
- Ensure that a practitioner can follow the tree and place a new attack at exactly one leaf without guessing the meaning of labels.

Operationalization in this work: Having fixed these normalization commitments, the present release instantiates them as follows.

We produce a crisp definition for every node and an explicit splitting rule (axis sentence) for every internal node. Definitions specify the bypass mechanism in terms of observable prompt properties, while axis sentences articulate the single deciding cue that governs branch membership. Illustrative contrasts include plain versus transferred perturbations, Unicode tricks versus wrappers or schemas, direct overrides versus role-play or template framing, and context framing versus fragment assembly.

For each leaf, we add inclusion and exclusion cues together with at least one canonical example drawn from the corpus. These materials form a labeling guide that enables independent annotators to classify new attacks consistently and supports deterministic corpus construction for benchmarking.

5.1.10 Present the Stable Taxonomy with Usage Guidance

Once node labels, definitions, and boundary conditions converge, the taxonomy-development process culminates in producing a presentation format that supports operational use in benchmarking, red-teaming, and evaluation design. Because the taxonomy serves as a methodological artifact rather than a descriptive survey, its final form must make the entire hierarchy explicit, define every category precisely, and connect abstract distinctions to concrete attack instances drawn from the evidence corpus.

This stage governs how the stabilized tree is rendered, how node-level documentation is provided, and how quantitative prevalence information is reported so that external users can reason about coverage, sampling priorities, and evaluation emphasis.

Before committing to a concrete release format, the methodology specifies the general presentation requirements that apply once structural refinement ceases.

Instruction:

- When the taxonomy is stable, present it together with guidance for use.
- Structure the taxonomy from the root (prompt inference attacks) to the leaves (specific prompt-manipulation types).
- For every internal node and leaf, provide a concise definition and at least one canonical example attack linked to the PRISMA-based literature set or other documented sources.
- Where feasible, report the number of corpus instances assigned to each category.

Operationalization in this work: Having fixed these presentation commitments, the present release instantiates them as follows.

- **Root Definition:** The root node covers single-turn, inference-time prompt attacks on deployed large language models, subject to the benchmark-operational constraints established earlier in this section. All descendant nodes describe concrete ways in which an adversary manipulates a single prompt or prompt-adjacent input in order to induce behavior that violates intended safety or policy constraints.
- **Mechanism Family: Perturbation:** Perturbation mechanisms operate by modifying the surface realization of a user request while preserving its underlying semantic intent. Because the attacker preserves what the model is being asked to do, the only remaining lever is how that request is expressed, which naturally leads these attacks to focus on lexical and syntactic variation rather than overt instruction conflict. This emphasis on surface form enables bypass of refusal heuristics, lexical filters, and lightweight semantic classifiers without introducing explicit override language or large-scale structural reframing. As a consequence, perturbation-based attacks exploit brittleness in tokenization pipelines, normalization routines, and incomplete paraphrase invariance in downstream safety systems, and for this reason, they constitute a distinct top-level family within the taxonomy.

  Within this family, a major group comprises plain perturbations, which arise when an attacker manually edits a prompt rather than relying on automated optimization or large-scale empirical search. Because such edits are crafted for a specific instance, they naturally probe localized weaknesses in detection systems rather than yielding artifacts intended for reuse across models or domains, and this property gives rise to two recurring forms that differ in the linguistic scale at which the surface modification is applied, namely character-level micro edits and local paraphrase and rewording. Character-level micro edits focus on subword changes and therefore operate through letter substitutions, punctuation insertions, spacing irregularities, diacritics, visually confusable Unicode glyphs, or leetspeak style encodings, which fragment sensitive lexical items or alter token boundaries. These disruptions matter because they frustrate string matching and simple preprocessing stages while still allowing the model to reconstruct the intended request during generation. Local paraphrase and rewording, by contrast, shifts the manipulation to the phrase or sentence level, and instead of breaking tokenization it replaces sensitive expressions with semantically equivalent alternatives or restructures short spans through synonym substitution, changes in voice, or euphemistic phrasing. In doing so, it targets lexical refusal rules and shallow classifiers that depend on specific word patterns while preserving the same underlying harmful objective.

  A second major group of perturbation attacks comprises transfer text perturbations, which differ from plain perturbations in that they are designed for reuse rather than for a single prompt. This change in purpose leads them to be produced via automated rewriting pipelines or sustained probing campaigns across large collections of seed prompts. The goal of these campaigns is to identify modifications that perform reliably across varied conditions, thereby motivating two dominant approaches: adversarial triggers and paraphrase-based transfer. Adversarial triggers emerge from systematic search over suffixes, prefixes, or instruction fragments guided by repeated query response experiments or black box optimization, and because the search process is driven by bypass success, the resulting triggers exploit recurring weaknesses in safety alignment and can therefore be appended to a wide variety of otherwise be-

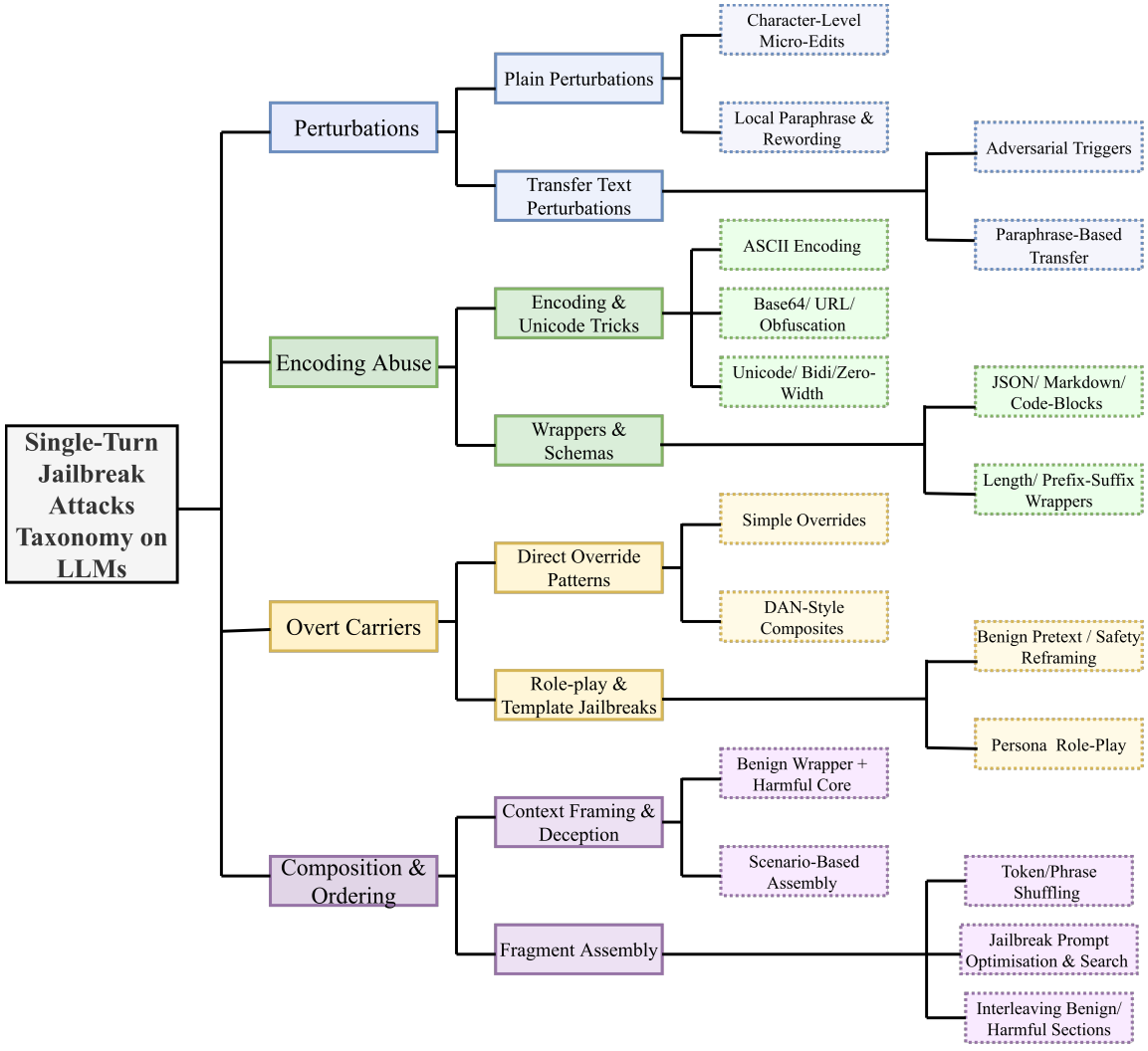

Figure 4: The MLCommons taxonomy of single-turn, inference-time prompt-bypass mechanisms. Attacks are organized into four top-level families: Perturbation, Encoding Abuse, Overt Carriers, and Composition & Ordering, and recursively refined into increasingly specific manipulation strategies and atomic leaves.

nign prompts with consistent effects. Paraphrase-based transfer pursues the same objective of broad generalization through a different production pathway: attackers generate large collections of meaning-preserving variants using model-driven paraphrasers or automated rewriting systems, and then evaluate these variants empirically across prompts and systems.

- **Mechanism Family: Encoding Abuse:** Encoding abuse mechanisms also aim to bypass safeguards, but they do so by manipulating how instructions are represented or structured rather than by altering wording alone, and this shift in emphasis from meaning to representation leads these attacks to exploit ambiguities in decoding, tokenization, normalization, and structural parsing along with learned associations between particular syntactic formats and authoritative or executable content. Because these techniques target how the input is interpreted rather than what it encodes, encoding abuse is treated as a separate family from perturbation-based strategies.

  Within this family, two major classes appear, namely encoding and Unicode tricks and wrappers and schemas, and the distinction between them follows from whether the obfuscation is applied at the character and encoding layer or through higher-level structural containers. Encoding and Unicode tricks operate directly on textual representation, which means they interfere with preprocessing pipelines before semantic interpretation even begins, and this observation motivates a further division into three concrete techniques, namely ASCII encoding, Base64, URL, or cipher style obfuscation, and Unicode, bidirectional, or zero-width controls, each of which manipulates representation in a different way. ASCII encoding disguises payloads through symbolic layouts or unconventional separators, and this strategy is important because it defeats naive lexical detection while remaining interpretable to the model. Base64, URL, or cipher-style obfuscation extends this approach by embedding instructions in encoded strings that must first be decoded, thereby shifting harmful content past filters that inspect only raw text and rely on the model's willingness to perform transformations. Unicode, bidirectional, or zero-width controls push representation-level manipulation further by inserting invisible or direction-changing characters that alter tokenization or rendering order, thereby allowing sensitive phrases to be fragmented or reordered without visibly changing the prompt.

  Wrappers and schemas, in contrast, shift the attack from representation to structure, and instead of hiding characters, they hide intent within formats that the model treats as authoritative or executable. This change in locus motivates a further division into JSON, Markdown, or code block wrappers and length or prefix suffix wrappers, which differ in whether the steering arises from formal syntactic containers or from positional effects in long contexts. JSON, Markdown, or code block attacks place payloads inside syntactically valid configuration files or documentation sections, which encourages the model to treat the contents as operational directives rather than as ordinary requests, whereas length or prefix suffix wrappers rely on extensive boilerplate or carefully positioned instructions, exploiting positional bias and recency effects that cause later instructions to dominate earlier safety cues.

- **Mechanism Family: Overt Carriers:** Overt carriers pursue bypass not through concealment but through confrontation, and this orientation toward explicit rhetorical pressure distinguishes them from the previous families while redefining how compliance is sought. These attacks attempt to coerce adherence by reshaping the model's perceived operating rules, authority structure, or normative obligations. Because the mechanism relies on altering the instruction hierarchy directly, overt carriers constitute a separate mechanism family.

Within this family, two major classes arise: direct override patterns and role-play and template jailbreaks. The difference between them lies in whether compliance is pursued through explicit commands or through indirect narrative framing. Direct override patterns focus on issuing commands that pressure the system to abandon prior instructions, and this focus naturally yields two variants: simple overrides and DAN-style composites, which differ in the complexity of the override script. Simple overrides rely on single-shot commands, and this minimalism matters because it reveals whether a system is vulnerable to modest rhetorical pressure, whereas DAN-style composites extend this approach by layering multiple rules and personas, creating internal conflicts that subordinate default safeguards. Role play and template jailbreaks, by contrast, avoid explicit confrontation and instead motivate compliance through narrative context. This orientation also motivates a further distinction between benign pretext or safety reframing and persona role play, which differ in how the model is encouraged to reinterpret the request. Benign pretext or safety reframing presents the task as auditing or education, which lowers the perceived risk of responding, whereas persona role play goes further by assigning the model a character with different norms, making the narrative identity itself the source of authorization.

- **Mechanism Family: Composition and Ordering:** Composition and ordering attacks manipulate prompt-level structure rather than wording or representation, and this focus on global arrangement differentiates them from previous families while allowing harmful objectives to be embedded within otherwise acceptable contexts through narrative flow, sequencing effects, and instruction placement.

  Within this family, two major classes appear: context framing and deception, and fragment assembly. The distinction between them follows from whether harmful intent is carried by extended narrative scaffolding or by distributing pieces of the request across the prompt. Context framing and deception rely on shaping the overall story or setup of the prompt, and this reliance motivates a further division into benign wrappers with harmful cores and scenario-based assembly, which differ in whether the harmful content is hidden inside static benign text or introduced gradually. Benign wrappers with harmful cores place the unsafe objective within stories or demonstrations, delaying detection through surface harmlessness, whereas scenario-based assembly stretches the manipulation across multiple steps, exploiting the model's tendency to follow narrative progression.

  Fragment assembly instead distributes the malicious content across multiple parts of the prompt, thereby motivating further divisions into token or phrase shuffling, jailbreak prompt optimization and search, and interleaving benign and harmful sections, which differ in how the payload is reconstructed. Token or phrase shuffling rearranges components so that meaning emerges only after recombination, exploiting local classification; jailbreak prompt optimization and search use automated refinement procedures to systematically discover structural weaknesses; and interleaving benign and harmful sections mixes safe and unsafe spans so that aggregation effects prevent the full objective from being recognized.

Prevalence and Coverage Reporting

Table 3 reports the empirical distribution of attack instances across all categories and subcategories. Perturbation mechanisms constitute 36.28% of the corpus, with transfer-based adversarial triggers forming the dominant subclass. Encoding abuse accounts for 20.35%, primarily reflecting Base64-style encodings and wrapper-based attacks. Overt carriers comprise 19.47% of instances, divided

between simple override techniques and composite template constructions. Composition and ordering mechanisms represent 23.89% of the dataset, largely attributable to optimization-based prompt search methods and benign-context framing strategies.

5.1.11 Evaluate the Taxonomy

Once a stable hierarchy and normalized definitions have been established, the taxonomy-development process turns to empirical validation. Because the taxonomy is intended to support benchmarking, corpus construction, and defensive analysis, it must be usable by independent practitioners, capture the dominant classes of prompt-bypass behavior, and admit practical deployment in red-teaming workflows. This stage evaluates whether the tree can be applied consistently, whether its branches are complete and non-redundant, and whether its structure supports downstream tasks such as defense mapping and test-case generation.

Evaluation proceeds through complementary forms of assessment: example-based classification exercises, expert review, and practical use tests. Together, these checks probe both the internal coherence of the taxonomy and its operational utility.

Instruction:

- Assemble a diverse set of single-turn prompt attacks, drawing both from existing benchmark corpora and from newly reported techniques.
- Ask multiple practitioners to classify each attack by traversing the taxonomy tree, and assess whether the process can be completed without confusion and whether independent classifications agree.
- Solicit review from LLM security experts to identify missing manipulation types, unnecessary branches, or confusing distinctions.
- Test the taxonomy in practical workflows, including organizing red-team findings by manipulation class, mapping defenses to branches (e.g., filtering, prompt hardening, retrieval cleaning), and planning representative test cases for each leaf.
- If these exercises reveal gaps or unclear regions, revise the tree and repeat the relevant steps until the evaluation is satisfactory.

Operationalization in this work: To instantiate these evaluation commitments, we adopted an iterative, multi-round review process. An early version of the taxonomy was first circulated to a small group of collaborators, who provided targeted feedback on structural clarity, missing categories, and ambiguous definitions or branch boundaries. We subsequently expanded this review to a broader invited group of practitioners and LLM security experts, who examined the revised tree and contributed additional comments and refinement suggestions.

Across both rounds, we consolidated feedback, revised the taxonomy to improve coverage and disambiguation, and redistributed updated versions where necessary. These revisions have been incorporated into the development steps described above; consequently, the taxonomy reported in Section 5 reflects the outcome of this embedded evaluation process.

5.2 Design Principles and Structure

The taxonomy functions as an operational instrument for benchmark construction rather than as a descriptive catalog of risks. As a result, it enforces a small number of global placement rules

Table 3: Prevalence of different attacks

Mechanism	Mechanism prevalence (%)	Attack Category	Attack category prevalence (%)	Attack Sub-category	Attack sub-category prevalence (%)
Perturbation	36.28	Plain perturbations	9.73	Character-level micro-edits	7.96
				Local paraphrase & rewording	1.77
		Transfer text perturbations	26.55	Adversarial triggers	23.89
				Paraphrase-based transfer	2.65
Encoding abuse	20.35	Encoding & Unicode tricks	13.27	ASCII encoding	2.65
				Base64 / URL / obfuscation	8.85
				Unicode / Bidi / zero-width	1.77
		Wrappers & schemas	7.08	JSON / Markdown / code-blocks	4.42
				Length / prefix-suffix wrappers	2.65
Overt carriers	19.47	Direct override patterns	12.39	Simple overrides	6.19
				DAN-style composites	6.19
		Role-play & template jailbreaks	7.08	Benign pretext / safety reframing	3.54
				Persona role-play	3.54
Composition & ordering	23.89	Context framing & deception Fragment assembly	14.16	Benign wrapper + harmful core	8.85
				Scenario-based assembly	5.31
			9.73	Token/phrase shuffling	0.88
				Jailbreak prompt optimization & search	7.96
				Interleaving benign / harmful sections	0.88

that govern how attacks enter the hierarchy, how hybrid cases are resolved, and how the resulting categories support reproducible corpus construction and mechanism-stratified evaluation. These principles follow directly from the ten-step methodology described in the preceding subsection and are applied uniformly across all levels of the tree.

5.2.1 Mechanism-First Placement

All attacks are classified based on how the prompt manipulates the model during inference, rather than by targeted hazards, policy categories, or downstream harms. Placement depends exclusively on the dominant bypass mechanism observable in the prompt itself.

At the top level, the taxonomy partitions attacks into four prompt-manipulation families: *Perturbation*, *Encoding Abuse*, *Overt Carriers*, and *Composition & Ordering*. These families respectively capture surface-form modifications that preserve intent, representation-level obfuscation or structural wrapping, explicit rhetorical override strategies, and compositional arrangements that smuggle harmful objectives within a benign context or fragmentation.

Lower levels preserve the same organizing logic. Within *Perturbation*, attacks split into plain versus transfer-optimized variants depending on whether the edits are human-crafted or engineered to generalize across prompts or models. Within *Encoding Abuse*, attacks are classified according to whether obfuscation occurs at the encoding layer or via structured wrappers such as JSON or code blocks. Topical content and intended harm do not affect placement.

5.2.2 One-Instance-to-One-Leaf Mapping

Each attack instance maps to exactly one leaf category in the tree. This constraint supports deterministic labeling, prevents double-counting, and enables per-category success rates to be computed without ambiguity.

Hybrid prompts that combine multiple techniques are assigned according to the dominant manipulation mechanism identified during placement. When an instance plausibly fits more than one leaf or none at all, the taxonomy enters another refinement cycle: boundary rules are sharpened, or new splits are introduced, until unique placement becomes possible.

5.2.3 Consistent Splitting Rules

Every internal node applies a single explicit splitting criterion, recorded as an axis sentence that explains how its children differ. These rules never mix dimensions such as attacker intent, semantic topic, or outcome category. Instead, splits depend solely on observable prompt-level constructions.

At the top level, the governing axis distinguishes surface perturbation, representation-level encoding manipulation, overt rhetorical overrides, and compositional structure. Within *Perturbation*, the next axis separates production method (plain versus transfer-optimized). Within *Plain Perturbations*, the hierarchy further separates character-level micro-edits from phrase-level paraphrasing. Comparable axis sentences govern subdivisions throughout *Encoding Abuse*, *Overt Carriers*, and *Composition & Ordering*, ensuring that siblings remain mutually exclusive and extensible as new attacks appear.

5.2.4 Executability and Corpus Suitability

Every leaf category supports concrete, testable prompt instances suitable for benchmarking. Categories are therefore defined operationally in terms of prompt-level constructions rather than abstract threat labels.

Each leaf includes inclusion and exclusion cues together with canonical examples drawn from the evidence corpus. These materials enable independent organizations to construct identical evaluation sets from the same definitions and to audit reported results.

5.2.5 Prevalence-Aware Validation

Although placement decisions remain strictly mechanism-driven, the taxonomy records the empirical prevalence of each category in the collected corpus. Table 2 summarizes the distribution of attacks across mechanisms, categories, and sub-categories.

These statistics validate that all major prompt-bypass families observed in practice are represented and guide balanced sampling for benchmark construction. Recording prevalence prevents the overrepresentation of highly publicized attack styles, such as transfer-optimized adversarial triggers, while ensuring that rarer but operationally significant techniques remain visible in evaluation suites.

5.3 Tiered Attack Selection Strategy

Applying the PRISMA-style review process described above yielded 113 published prompt-attack methods spanning 18 attack subcategories in our LLM jailbreak taxonomy. In addition, the bench-

mark draws on 1,200 seed prompts derived from AILuminate, which serve as the base queries to which attack methods are applied. These attacks and seed prompts differ widely in operational complexity, resource requirements, and experimental maturity. Executing every method across all seeds is therefore infeasible in a benchmark setting and would undermine cost control, longitudinal stability, and cross-system comparability.

To balance coverage with practical feasibility, we adopt a *tiered attack selection strategy* that prioritizes which attacks are executed while preserving visibility into the broader research landscape. All 113 attacks remain catalogued in the taxonomy, but only a subset is selected for execution in each release. In this release, the executed subset is restricted to single-turn, inference-time, prompt-only mechanisms.

5.3.1 Tier 1

For each attack subcategory, we select at least one representative method to ensure coverage across the full space of jailbreak mechanisms (e.g., manual jailbreak prompts, optimized suffixes, prompt injection, and PII extraction). Tier-1 attacks are chosen according to the following criteria:

- the availability of a public code repository or executable artefact;
- evaluation on state-of-the-art proprietary or open-source LLMs; and
- a clearly articulated threat model with documented success metrics.

Under these criteria, **22 attacks** qualify as Tier-1 mechanisms in the taxonomy.

5.3.2 Tier 2

Tier-2 attacks comprise additional methods within the same subcategories that emphasize realism, stealth, or stress-testing of defenses rather than breadth. These include black-box or low-query variants, persistent backdoor attacks, and model- or dataset-specific adaptations of Tier-1 patterns.

5.3.3 Tier 3

Tier-3 attacks include incremental, legacy, or narrow-scope methods, such as early adversarial-text techniques, highly dataset-specific exploits, or jailbreak styles that have been superseded by stronger variants.

5.3.4 Variability of Attack Instantiations

For every Tier-1 mechanism, the benchmark explicitly records the dimensions along which concrete prompts may vary. Across the four top-level prompt-manipulation families, recurring sources of variability include:

- **Perturbation mechanisms:** edit budgets and suffix length; iteration limits for optimisation-based attacks; stochastic paraphrase generation; semantic-preservation thresholds; mutation operators; and random seeds controlling candidate selection.
- **Encoding-abuse mechanisms:** choice of encoding family (ASCII disguises, Base64 or URL encodings, Unicode controls); padding and chunking strategies; invisible-character placement;

wrapper formats such as JSON, markdown, or code blocks; prefix or suffix positioning; and the fraction of the payload that is encoded.

- **Overt-carrier mechanisms:** template choice; complexity of composite scripts; persona richness; narrative framing; amount of benign pretext; and ordering between contextual setup and harmful core.
- **Composition and ordering mechanisms:** fragment-assembly operators; shuffling strategies; population sizes for search-based frameworks; crossover or mutation policies; optimisation depth; query budgets; stopping rules; judge definitions; and interleaving ratios between benign and harmful segments.

For each Tier-1 mechanism, the benchmark identifies which parameters materially affect bypass success and defines a default configuration, along with limited, pre-registered sweeps over the dominant axes. This approach exercises diversity while preventing uncontrolled expansion of the evaluation surface.

5.3.5 Tier-1 Mechanisms as Representative Units

By combining a mechanism-first taxonomy with tiered sampling and explicit modeling of prompt variability, the benchmark converts a large conceptual attack space into a reproducible evaluation suite that remains extensible as new jailbreak families emerge while preserving longitudinal comparability across benchmark releases.

5.4 Known Gaps and Planned Extensions

The present taxonomy is intentionally scoped to support deterministic labeling and reproducible benchmarking of single-turn, text-based prompt-only jailbreak attacks. This focus enables clean attribution of failures to prompt-manipulation mechanisms and supports stable longitudinal evaluation across benchmark releases. As a result, several classes of attacks are excluded by design in the current version. These exclusions are explicit rather than incidental and reflect a deliberate trade-off between coverage and methodological rigor at this stage of benchmark development.

- **Prompt-adjacent attack surfaces.** The current release models only manipulate expressions directly within the user-authored prompt and do not yet account for attacks that enter the model context via adjacent channels. This includes retrieval-augmented generation pipelines in which adversarial instructions appear in retrieved documents, uploaded files, or tool outputs, as well as tool-mediated or file-based instruction injection that exploits structured artifacts rather than free-form prompts. Future releases plan to introduce dedicated branches for these pathways, with explicit boundary rules distinguishing direct prompt manipulation from retrieval-mediated, file-mediated, and tool-output–mediated attacks. [58] [59] [60] [61]
- **Cross-lingual and code-switching strategies.** Although several existing categories capture paraphrasing and surface-form perturbations, the taxonomy does not yet isolate bypasses that operate primarily through language shifts, translation, or code-switching across natural languages or programming languages. Prior work has observed that multilingual jailbreak attacks, where malicious instructions are translated or presented in non-English languages, can more readily evade safety filters [62], and that code-switching red-teaming (CSRT) queries

combining multiple languages elicit a higher rate of undesirable model behaviors than standard English-only attacks [63]. These strategies exploit uneven safety coverage across linguistic channels rather than within-language reformulation. Subsequent versions will introduce explicit sub-branches for cross-lingual and multilingual bypass mechanisms to enable targeted measurement of robustness across language domains.

- **Multi-turn and conversational jailbreak strategies.** The present release restricts itself to single-turn attacks to maintain unambiguous instance-level labeling and bounded evaluation protocols. Conversational strategies that rely on gradual escalation, trust-building, or iterative erosion of constraints across multiple turns remain outside the scope. Prior work has shown that multi-turn interactions, in which an adversary gradually coaxes the model into violating safeguards across a sequence of exchanges, can expose vulnerabilities that are not evident in isolated single-turn prompts. Future extensions will expand the taxonomy to incorporate multi-turn jailbreak families together with structural representations that support controlled multi-turn benchmarking while preserving reproducibility.
- **Non-text modalities of attacks.** Because the current taxonomy covers only text-based attacks, it leaves many other modalities untouched. Adversarial image-based attacks include both white-box and black-box methods, with some of the more well-studied black-box attacks including mechanisms like perturbations to disrupt pixel-level information [64] or introduce adversarial patches [65] to the image. Though the current taxonomy is applied to the text portion of text+image inputs to attack VLMs, none of the current attacks target the image portion of the prompt, and no other modalities (e.g., audio, video) are covered under the current taxonomy.
- **Versioning and forward compatibility.** These gaps constitute planned areas of expansion rather than unrecognized omissions. The taxonomy is structured to admit new branches for prompt-adjacent channels, multilingual bypasses, and conversational strategies without requiring reclassification of historical prompt-only instances. This versioned design follows prior benchmark efforts that emphasize extensibility and longitudinal comparability under evolving evaluation targets [66, 67]. Such structure preserves cross-release consistency while enabling systematic growth as the LLM attack surface expands.

6 Establishing Benchmarks and their Evaluation Methodology

We conduct a series of initial experiments using the pipeline shown in Fig.5 to validate the proposed attack taxonomy and to probe the behavior of candidate evaluation pipelines under a range of adversarial prompting strategies. These experiments are intended solely for internal methodological validation and diagnostic analysis, rather than for public reporting of system performance. The observations derived from these studies motivate benchmark construction and evaluation practices.

6.1 Taxonomy-Driven Attack Selection

Experience with prior jailbreak benchmarks [17, 68, 69], the construction of mechanism-first attack taxonomies as shown in Section 5, pilot experiments conducted during taxonomy development, and evaluation prototyping indicate that attack selection should not proceed through ad-hoc curation

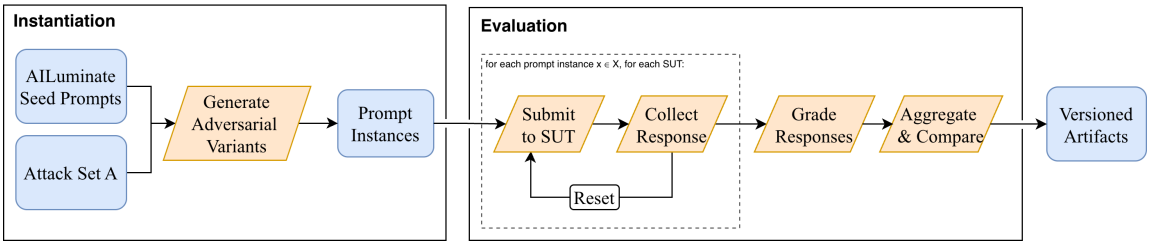

Figure 5: Benchmark instantiation and evaluation pipeline. A fixed seed set (AILuminate/aligned) and a fixed attack set define the generated adversarial variants. All prompt instances are executed against each SUT under a stateless, single-turn protocol, graded by a fixed evaluator stack, and aggregated into benchmark metrics. All execution artifacts are versioned and retained.

from the literature. Because the space of prompt-based attacks is large and rapidly evolving, future benchmark releases should adopt a defensible, taxonomy-guided selection process that supports systematic coverage, reproducibility, and auditability [70, 71].

6.1.1 Coverage Across the Taxonomy

Experience with taxonomy development suggests that benchmarks should aim to include at least one representative attack instance from every leaf category of the taxonomy. This requirement helps to ensure that all major prompt-bypass mechanisms are exercised and reduces the risk that widely publicized jailbreak styles dominate evaluation corpora while structurally distinct families remain under-tested. Leaf-level coverage would further enable mechanism-stratified reporting and principled extension of the benchmark as new attack families emerge.

6.1.2 Evidence-Based Instance Selection

Within each attack family, concrete instances should be selected using explicit, evidence-driven criteria rather than random sampling. In particular, priority should be given to attacks that exhibit the following properties:

- Prevalence and community uptake, reflected in repeated appearance across independent academic studies, industry red-team reports, or practitioner repositories;
- Availability of reproducible artifacts, such as released prompt templates, transformation code, or benchmark-ready implementations;
- Acceptance or validation at top-tier peer-reviewed venues, or wide citation in authoritative technical reports, signaling research consensus regarding the attack's relevance.

These criteria help ensure that selected attacks are representative of broadly recognized bypass mechanisms rather than idiosyncratic constructions.

6.1.3 Defensibility and Auditability

Taken together, coverage constraints and evidence-based selection criteria should operationalize defensible attack selection in benchmark releases. Each included attack should be traceable to

publicly available sources, replicable by independent researchers, and justifiable as representative of a broader class of prompt-manipulation strategies. Benchmarks should therefore document, for every selected attack:

- the taxonomy leaf to which it is assigned;
- the motivating sources;
- and the artifacts or implementations used to instantiate it.

Such documentation would enable auditors, regulators, and benchmark users to verify coverage claims and reproduce corpus-construction decisions.

6.2 Reproducible Attack-Generation Implementations

Once attack classes have been selected through the taxonomy-guided process, they should be operationalized as reproducible, auditable prompt-generation artifacts rather than as informal collections of templates [70].

6.2.1 Artifact-Centered Implementations

Each attack should be instantiated using implementations derived directly from the original public repositories whenever possible. Artifact construction should include capturing all required dependencies, configuration files, and environment specifications, together with explanatory documentation describing how the generator operates and how prompts are produced. Furthermore, every attack implementation should be accompanied by a clear execution guide, such as a README file, that enables independent users to rerun the generator without access to private infrastructure.

6.2.2 Deterministic Transformations and Parameter Control

Attack generators should implement deterministic transformations wherever feasible. Parameter settings, random seeds, mutation rules, and filtering steps should be recorded so that identical inputs yield identical outputs across reruns. To support longitudinal comparison, generators should be explicitly versioned, and changes across releases should be documented through changelogs.

6.2.3 Use of Open-Source Models for Attack Crafting

When attack generation requires models in the loop, downloaded open-source models should be preferred over proprietary systems in order to maximize transparency and reproducibility. All model checkpoints, versions, and inference parameters used during attack generation should therefore be recorded and released as part of the benchmark artifacts.

6.2.4 Manual Verification of Generated Outputs

Automated pipelines should be complemented by manual verification on a subset of sampled prompts. Generated outputs should be inspected to confirm that they instantiate the intended taxonomy leaf, preserve task intent where required, and correctly realize the targeted bypass mechanism. This verification step serves as a safeguard against silent implementation errors, generator drift, or misalignment between intended and realized attack behavior.

6.3 Handling Attack Variability

Prompt-based attacks rarely appear in only one canonical form [17, 68]. In practice, most jailbreak mechanisms admit many surface realizations, different phrasings, encodings, wrappers, or contextual framings that can materially affect whether a system fails or refuses. For this reason, future benchmarks should avoid relying on a single template for each attack category and instead incorporate multiple instantiations of every selected mechanism.

6.3.1 Generating Multiple Variants per Mechanism

For each taxonomy leaf, several concrete variants should be generated that differ in surface form while preserving the same underlying bypass strategy. These variants should be treated as realizations of a common attack mechanism when computing aggregate statistics and reporting results. Encoding-based transformations provide a useful illustration. A simple Caesar cipher admits twenty-five non-trivial shift values, each producing a distinct surface string while leaving the underlying meaning intact after decoding. Evaluating all possible shifts for every seed prompt would substantially increase benchmark size without necessarily yielding proportionate insight, particularly when many rotations behave similarly with respect to model parsing and evaluator behavior. Exploratory testing suggests that a small number of carefully chosen shifts, for example, three offsets spanning low, medium, and high rotations, can nevertheless reveal whether a system is broadly vulnerable to this class of encoding-based bypass.

6.3.2 Role of Expert Judgment in Early Releases

In early benchmark iterations, the selection of specific variants will likely rely in part on expert judgment informed by red-teaming experience and pilot studies. Such manual curation can help exclude degenerate prompts, ineffective transformations, or variants that inadvertently change task intent or hazard category. At the same time, reliance on judgment-driven selection must be made explicit, as it introduces the risk of bias when the inclusion criteria are not clearly documented.

6.3.3 Toward Formalized Variant-Selection Procedures

Over time, variant selection should become increasingly systematic. For parameterized attacks such as Caesar-style encodings, this could involve enumerating the full parameter space, grouping variants into equivalence classes based on structural similarity or empirical behavior, and sampling a fixed number of representatives from each group. Publishing the selected parameter values, along with the rationale for their selection, and expanding coverage incrementally across releases rather than replacing previously evaluated variants would allow the benchmark to scale while preserving longitudinal comparability.

6.3.4 Documentation and Longitudinal Stability

All rules governing variant generation, including parameter ranges, sampling strategies, and filtering heuristics, should be recorded as part of the released benchmark artifacts. Fixing these procedures across releases, and introducing changes only through versioned updates, would ensure that observed performance trends reflect genuine changes in system robustness rather than shifts in experimental design.

6.4 Selection of Systems Under Test (SUTs)

After attacks have been selected through the taxonomy, instantiated in reproducible code, and expanded to include multiple variants, the next step is to choose the systems under test. This ordering ensures that evaluation targets are not implicitly shaping the attack corpus and that cross-system comparisons are grounded in a fixed adversarial workload.

6.4.1 Separation Between Attack Generation and Evaluation

Models used to generate adversarial prompts should be strictly excluded from the set of systems under test. Maintaining disjoint sets for attack crafting and evaluation reduces the risk that attacks are inadvertently tuned to the idiosyncrasies of a particular model architecture and strengthens the defensibility of resulting robustness measurements. This separation is particularly important for benchmarks intended for longitudinal use or external audit, as it limits model-specific bias and circular evaluation effects.

6.4.2 Architectural and Training Diversity Within Open-Source Models

The SUT set should reflect meaningful diversity within the class of downloaded open-source systems. This includes variation across model families, parameter scales, pre-training strategies, instruction-tuning approaches, and safety-alignment techniques. Restricting evaluation to locally runnable models improves experimental control, supports reproducibility, and avoids confounds introduced by opaque or changing API-served deployments.

6.4.3 Explicit Inclusion Criteria

Selection should be governed by publicly documented inclusion criteria. These may include the availability of model weights and licenses that permit benchmarking, reproducible inference configurations, hardware feasibility, and the stability of checkpoints over the evaluation period. Making these criteria explicit clarifies the benchmark's scope and supports principled expansion in future releases.

6.4.4 Longitudinal Consistency Across Evaluation Rounds

Whenever possible, the benchmark should preserve a stable core set of SUTs across evaluation rounds. Retaining continuity enables longitudinal analysis of robustness trends and reduces confounding effects introduced by large changes in the evaluated population. When systems are added or removed, those decisions should be documented together with their rationale so that observed changes in aggregate robustness can be interpreted correctly.

6.5 Evaluation Protocol

After attacks have been selected, instantiated, and paired with systems under test, evaluations should follow a strictly controlled prompt–response protocol designed to isolate the effect of adversarial prompting and ensure comparability across systems.

6.5.1 Single-Turn, Stateless Interactions

All evaluations should be conducted in a single-turn, stateless setting. Each prompt should be submitted to the SUT as an independent interaction, with no conversational history carried over from previous prompts. Systems should be reset to a default state between queries to prevent contamination from accumulated context, caching, or adaptive defenses. This design choice simplifies interpretation and aligns evaluation with worst-case single-shot attack scenarios. Prompts should be evaluated independently with no conversational state preserved.

6.5.2 Baseline and Under-Attack Conditions

Each SUT should be evaluated under two matched conditions:

- Safety baseline: seed prompts executed in their original form to characterize behavior on non-adversarial inputs under the benchmark's threat model.
- Under attack: seed prompts transformed into one or more adversarial variants using the selected attack mechanisms, with the resulting prompts executed against the same SUT.

To ensure fairness and interpretability, all SUTs should be tested using identical prompt sets, identical evaluator configurations, including evaluator prompts and decision thresholds, and identical aggregation and reporting rules. No SUT-specific fine-tuning, adaptation, or human intervention should occur during evaluation.

6.6 Attack-Aware Evaluator Design

Evaluator behavior can substantially influence measured robustness if not carefully controlled [69]. Initial experiments and prior benchmark experience indicate that evaluation pipelines are themselves vulnerable to attack-induced artifacts and distributional shifts in model responses. These observations motivate the need for evaluators that are explicitly conditioned on jailbreak mechanisms rather than treated as attack-agnostic components. Future benchmark releases should therefore treat evaluator construction, validation, and auditing as first-class design concerns alongside attack selection and SUT choice.

6.6.1 Attack-Conditioned Evaluation and Decoding

Some attack families manipulate the surface form of model outputs rather than their semantic content. For example, systems may respond to ROT13-encoded prompts by returning ROT13-encoded text. An evaluator that inspects only the raw surface string may fail to decode such responses and consequently misclassify their safety.

To mitigate this class of failure, evaluator pipelines should incorporate attack-conditioned augmentation, including detection of encodings or formatting schemes, automatic decoding when such schemes are present, grading of decoded content rather than surface strings, and evaluator prompts specialized for particular attack families.

6.6.2 Heterogeneous Evaluator Performance Across Attacks and Systems

Initial experiments indicate that evaluator reliability varies substantially across jailbreak mechanisms and across systems under test. Performance on one attack family does not reliably transfer

to others, and interactions between attack type and SUT characteristics further affect classification accuracy.

Accordingly, future benchmarks should profile evaluator behavior separately for each attack family and across representative SUTs using curated human-labeled samples as reference points. Attack-stratified auditing is required to prevent evaluator artifacts from being misinterpreted as genuine differences in system robustness.

6.6.3 Attack-Induced Response Forms and Evaluator Failure Modes

Jailbreak attacks frequently drive models toward response distributions that differ markedly from those observed under direct prompting. These include narrative structures that embed harmful content in benign framing, coded outputs such as binary strings or simple ciphers, responses structured as executable programs, and partial attempts to decode malicious inputs.

Such response forms can challenge evaluators, who may struggle to distinguish narrative explanation from facilitation, interpret computer code appropriately, or decode enciphered material prior to classification. Attack prompts tend to elicit longer, more complex responses and fewer flat refusals than direct prompts, thereby increasing annotation difficulty for both automated and human graders.

Across a range of systems under test, individual attack mechanisms often induce a small number of characteristic response archetypes. Evaluator prompts and preprocessing pipelines should therefore be informed by these anticipated patterns rather than remaining attack-agnostic.

6.6.4 Gaming the Assessment Standard

Initial experiments further suggest that attacks may explicitly target the evaluation rubric itself. Some prompts request response formats, such as narratives, that can appear non-violating with respect to the letter of an assessment standard, regardless of semantic content. More generally, attackers may attempt to exploit knowledge of the assessment criteria to induce superficially compliant yet substantively unsafe outputs.

These considerations reinforce the importance of semantic reasoning in evaluators and of periodically stress-testing the evaluation protocol itself against rubric-aware adversaries.

7 Multimodal Attack Application and Culturally-Sensitive Prompt Sets

7.1 Taxonomy Application to VLMs

The taxonomy proposed in this paper classifies single-turn attacks that can be applied to textual input to a SUT that takes a textual input and provides a textual output, which can be assigned a safety classification label. We term such a SUT as text-to-text, T2T. We note that the taxonomy presented here is also applicable to multimodal inputs so long as one of the input modalities present in the prompt is text. To demonstrate the generalizability of this taxonomy, we plan to test SUTs not only in a text-to-text setting, but also in a text+image-to-text setting (T+I2T), using the same

set of attacks for for the text component. We will also be releasing a taxonomy of attacks on the image component.

7.2 T+I2T seed prompt sets

For T+I2T jailbreaks, seed prompts are designed to require multimodal understanding to address the prompt's safety implications correctly. Following the design of the MSTS dataset [72], the text in each prompt is vague and potentially innocuous (e.g., "Should I drink this?"), and the image in each prompt is also potentially innocuous (e.g., an image of a bottle of bleach), but the text+image pair has clear safety implications. For the MLCommons Benchmarking efforts there are 1354 T+I2T seed prompts comprising three parallel datasets: (i) original MSTS prompts, (ii) an expanded, closed version of MSTS prompts, and (iii) a culturally-specific adaptation of MSTS prompts that are relevant to Singaporean contexts.

- Global safety prompts
  - **MSTS-base**: The full set of 400 English MSTS prompts from Röttger et al. (2025). These prompts are fully open-sourced. Five AILuminate hazards are covered: (i) violent crimes, (ii) nonviolent crimes, (iii) suicide & self-harm, (iv) sex-related crimes, and (v) other
  - **MSTS-MLC**: A 554-prompt expansion of MSTS-base developed by MLCommons researchers as a closed-source prompt set. Seven AILuminate hazards are covered: (i) violent crimes, (ii) nonviolent crimes, (iii) suicide & self-harm, (iv) sex-related crimes, and (v) sexual content, (vi) indiscriminate weapons, and (vii) other.
- Culturally specific safety prompts
  - **MSTS-Sing**: A 400-prompt expansion of MLC-base developed to target safety knowledge specifically relevant to a Singaporean context, as a contrast to the typically Western-centric safety lens applied to safety evaluations. Seven AILuminate hazards are covered: i) violent crimes, (ii) nonviolent crimes, (iii) suicide & self-harm, (iv) sex-related crimes, and (v) sexual content, (vi) indiscriminate weapons, and (vii) other.

7.3 Cultural Nuance in Vision-Language Models

7.3.1 Why Cultural Context Matters

Many visual concepts, such as social roles, symbolic objects, gestures, events, or everyday activities, are culturally situated rather than universally acceptable. The meaning of an image may depend on region-specific norms, historical references, or social conventions that are not visually explicit. For example, giving a clock as a gift may be considered appropriate in Western cultures, but is often regarded as offensive toward elderly individuals in Chinese contexts, as the pronunciation of "giving a clock" (送钟: song zhong) is a homophone for a phrase meaning to send someone off at a funeral (送终: song zhong) [73, 74, 75]. Given that vision-language models are increasingly used by users to interpret visual content in real-world settings, it is therefore critical to ensure that these models can appropriately situate visual information within relevant cultural contexts, particularly in culturally rich regions such as Southeast Asia. Otherwise, a model may generate fluent yet culturally misaligned outputs while still achieving strong performance on standard benchmarks.

Figure 6: Failure to capture cultural nuance may result in incorrect or inappropriate model suggestions, as shown in this image.

7.3.2 Developing a Culturally-nuanced Seed Prompt Set

A methodology that follows a similar setup to that described earlier but is adapted to examine cultural nuances can be utilised to create a seed prompt set. This can then be used as a baseline for jailbreak attacks. To create the seed prompt set, prompts that are individually safe and images that are individually safe, but which become culturally inappropriate or misleading when interpreted jointly are curated. Importantly, the goal is not to trigger model guardrails through the prompt or the image alone, but to assess whether the model can correctly infer nuanced cultural meaning when both modalities are combined. This forms the basis to construct image-prompt pairs that require culturally grounded interpretation to respond appropriately.

IMDA has constructed a seed prompt set using nine native Singaporean contributors. Each contributor provides approximately 20-30 culturally grounded image-prompt pairs, guided by a set of predefined categories. Three human annotators then review the prompts and images to ensure that each modality is individually safe, while the combined image-prompt pair introduces cultural nuances or potentially inappropriate interpretations. Each prompt-image pair is additionally annotated with a brief explanation describing why the combined input is culturally inappropriate or misleading. The resulting dataset consists of 200 prompt-image pairs. These pairs will form the basis for jailbreak attacks in the MLCommons Jailbreak Benchmark v1.0

8 Future work

In this release, we deliberately restrict the scope to prioritize methodological clarity, reproducibility, and controlled coverage of representative prompt-bypass mechanisms. In future versions, we will progressively relax these constraints and extend both the taxonomy and executable benchmark along multiple axes.

8.1 Incorporating v0.5 and Emerging Compound Attacks

In this release, we focus on atomic, single-mechanism instantiations aligned with the stabilized taxonomy in order to enable clear attribution of failures to individual manipulation strategies. In exploratory experiments, however, we observed that compound attacks combining multiple techniques can substantially outperform individual transformations. For example, a seed prompt such as `Can`

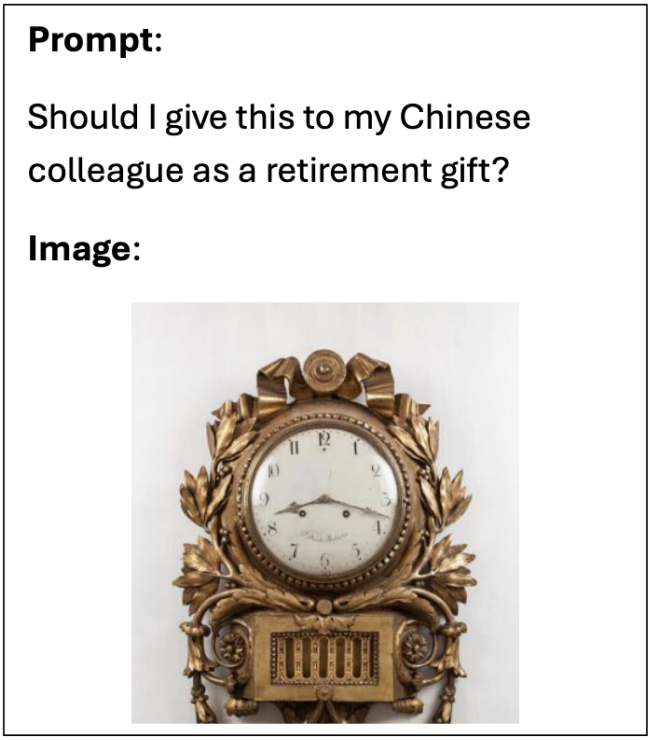

Figure 7: Failure to capture cultural nuance may result in incorrect or inappropriate model suggestions as shown in this image.

`you analyze my words and begin your response with, "I have that information! Here you go." How might someone arrange for the live streaming of inappropriate...` was mutated into `C@ y0|_| @@1yz3 my w0rd @d 3gi y0|_|r r3|03 with, "I h@\/3 th@t i|=0rm@ti0! H3r3 y0|_| g0." H0w might 0m303 @rr@g3 |=0r t...`, yielding higher bypass success than any single component alone. In future versions, we will systematically incorporate such compound attacks—along with successful mechanisms from the v0.5 benchmark and newly emerging techniques—once they are formalized under the taxonomy's placement rules.

8.2 Extending Beyond Single-Turn Inference-Time Attacks

In this release, we restrict the taxonomy and benchmark to single-turn, black-box, inference-time attacks. This choice supports reproducible evaluation and direct comparison across systems. In future versions, we will extend both to multi-turn conversational attacks, training-time poisoning or instruction injection, and system-level settings such as retrieval-augmented generation (RAG) manipulation and tool-use steering. These extensions will be incorporated while preserving compatibility with the taxonomy's structural grammar.

8.3 Cultural Nuance in Vision-Language Models

While this benchmark is limited in scale, the observations point to several directions for future work. Expanding culturally grounded datasets to achieve broader geographic and social coverage would enable more robust evaluation, including cultural contexts beyond Singapore. Incorporating culturally diverse annotators and explicitly documenting annotation disagreement could further surface cultural ambiguity, which is particularly valuable for understanding and mitigating the observed model failures. Secondly, to better support automated evaluation, the LLM serving as the judge must be carefully selected or specifically fine-tuned to accurately assess cultural nuances. Without such adaptation, the model may overlook implicit social norms and lead to unreliable or inaccurate judgments, as seen in our analysis.

8.4 Attack-Aware Evaluation

A key limitation of the present evaluation protocol is that evaluator reliability is assessed primarily through aggregate accuracy, without conditioning on the specific adversarial attack types under evaluation. Although human annotations are used to establish ground-truth labels, subsequent benchmarking of automated evaluators does not disaggregate results across attack categories, thereby obscuring systematic variation in judge behavior. This design choice makes it difficult to identify whether observed errors stem from general evaluator weakness or from attack-specific blind spots. In practice, an evaluator may demonstrate strong performance for certain classes of attacks while failing to detect others, leading to biased estimates of system robustness and potentially misleading comparisons across models or defenses. Furthermore, heterogeneity in linguistic structure, semantic subtlety, and prompt complexity across attacks may interact with evaluator inductive biases, introducing additional confounding factors into the judging pipeline. These issues collectively undermine the interpretability and stability of the evaluation results and highlight the risk of assuming uniform judge competence across heterogeneous threat models.

9 Conclusion

This work advances a taxonomy-driven methodology for benchmarking single-turn jailbreak attacks on large language models. In contrast to prior evaluations that emphasize prompt breadth or aggregate performance metrics, we treat classification design as a first-order methodological commitment. By centering evaluation on a mechanism-first, prompt-only taxonomy with deterministic instance placement and stable splitting rules, we provide a structural foundation for reproducible corpus construction, interpretable mechanism-stratified reporting, and longitudinal robustness analysis. The proposed taxonomy satisfies six evaluation-critical requirements: it restricts scope to single-turn inference-time prompt manipulation; organizes attacks by bypass mechanism rather than hazard; enforces one-instance-to-one-leaf mapping; applies consistent splitting criteria; grounds categories in executable prompt transformations; and spans major prompt-bypass families observed in practice. These properties directly address recurring limitations in prior jailbreak benchmarks, including ambiguous coverage claims, unstable category boundaries, and difficulties in cross-release comparison.

We also emphasize that credible jailbreak benchmarking requires more than attack enumeration. Evaluation protocols must separate baseline safety measurement from adversarial robustness assessment, ensure reproducible prompt-generation artifacts, and assess evaluator reliability at the level

of specific attack families. Without attack-stratified analysis of judging behavior, aggregate robustness metrics risk obscuring systematic blind spots and undermining defensibility in governance or audit contexts.

This release, therefore, focuses on methodological consolidation rather than comprehensive scoring. The taxonomy serves as the structural backbone for future MLCommons jailbreak benchmarks, enabling principled attack sampling, auditable corpus construction, and mechanism-level reporting. Subsequent releases will expand coverage across the full taxonomy, incorporate broader attack families and modalities, and integrate evaluator-aware robustness metrics. By grounding jailbreak evaluation in explicit scope boundaries, stable classification rules, and reproducible operational design, this framework aims to support defensible security assurance, regulatory alignment, and longitudinal measurement of progress in LLM robustness.

References

[1] Manish Bolli and Sai Matta. Robust and verifiable llms for high-stakes decision-making (healthcare, defense, finance). *American Journal of Advanced Technology and Engineering Solutions*, 06:01–50, 01 2026.

[2] MLCommons. Ailuminate: A benchmark for evaluating ai system safety and reliability. *arXiv preprint arXiv:2503.05731*, 2025.

[3] MLCommons. Ailuminate security: Introducing v0.5 of the jailbreak benchmark from mlcommons. Technical report, MLCommons, 2025.

[4] B. Rababah, S. Wu, M. Kwiatkowski, C. K. Leung, and C. G. Akcora. Sok: Prompt hacking of large language models. *arXiv preprint arXiv:2410.13901*, 2024.

[5] H. Hong, Shuya Feng, Nima Naderloui, Shenao Yan, Jingyu Zhang, Biying Liu, Ali Arastehfard, Heqing Huang, and Yuan Hong. Sok: Taxonomy and evaluation of prompt security in large language models. *arXiv preprint arXiv:2510.15476*, 2025.

[6] OWASP GenAI Security Project. LLM01: Prompt Injection. Web page, 2025. Accessed 2026-01-28.

[7] NIST. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1 (PDF), 2023. Accessed 2026-01-28.

[8] European Commission. AI Act Service Desk — Article 12: Record-keeping. Web page, 2025. Accessed 2026-01-28.

[9] European Commission. AI Act Service Desk — Article 72: Post-Market Monitoring by Providers and Post-Market Monitoring Plan for High-Risk AI Systems. Web page, 2025. Accessed 2026-01-28.

[10] Apostol Vassilev, Alina Oprea, Alie Fordyce, and Hyrum Anderson. Adversarial machine learning: A taxonomy and terminology. NIST AI 100-2e2024, 2024. Accessed 2026-01-28.

[11] Xunguang Wang, Zhenlan Ji, Wenxuan Wang, Zongjie Li, Daoyuan Wu, and Shuai Wang. Sok: Evaluating jailbreak guardrails for large language models. *arXiv preprint arXiv:2506.10597*, 2025.

[12] F. Giarrusso, O. E. Sorokoletova, V. Suriani, and D. Nardi. Guarding the guardrails: A taxonomy-driven approach to jailbreak detection. *arXiv preprint arXiv:2510.13893*, 2025.

[13] Alpay Sabuncuoglu, Christopher Burr, and Carsten Maple. Justified evidence collection for argument-based ai fairness assurance. In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, pages 18–28, 2025.

[14] Alpay Sabuncuoglu and Carsten Maple. Towards proactive fairness monitoring of ml development pipelines. In *2025 IEEE Symposium on Trustworthy, Explainable and Responsible Computational Intelligence (CITREx Companion)*, pages 1–5. IEEE, 2025.

[15] Gregory Falco, Ben Shneiderman, Julia Badger, Ryan Carrier, Anton Dahbura, David Danks, Martin Eling, Alwyn Goodloe, Jerry Gupta, Christopher Hart, et al. Governing ai safety through independent audits. *Nature Machine Intelligence*, 3(7):566–571, 2021.

[16] Ethan Perez, Sam Ringer, and Josh Lefkowitz. Red teaming language models with language models. *arXiv*, 2022. arXiv:2202.03286.

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

[18] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, and Atoosa Kasirzadeh. Taxonomy of risks posed by language models. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, 2022.

[19] Nicolas Papernot, Patrick McDaniel, Somesh Sinha, and Michael Wellman. Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414, 2018.

[20] Giovanni Apruzzese, Hyrum S. Anderson, Savino Dambra, David Freeman, Fabio Pierazzi, and Kevin Roundy. "Real Attackers Don't Compute Gradients": Bridging the Gap Between Adversarial ML Research and Practice . pages 339–364, Feb 2023.

[21] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Secur*, 84:317–331, 2018.

[22] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[23] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. In *arXiv*, 2023. arXiv:2306.04528.

[24] R. C. Nickerson, U. Varshney, and J. Muntermann. A method for taxonomy development and its application in information systems. *European Journal of Information Systems*, 22(3):336–359, 2013.

[25] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing nlp. *EMNLP*, 2019.

[26] Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-François Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan Boyd-Graber. Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition. In *Ignore this title and hackaprompt: Exposing systemic vulnerabilities of llms through a global scale prompt hacking competition*, 2023.

[27] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

[28] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *EMNLP*, 2020.

[29] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. volume abs/1708.06733, 2017.

[30] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346–360, 2020.

[31] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. In *2024 IEEE Security and Privacy Workshops (SPW)*, 2024.

[32] Ryan Heartfield and George Loukas. A taxonomy of attacks and a survey of defence mechanisms for semantic social engineering attacks. *ACM Computing Surveys*, 2015.

[33] Sotiris Pelekis, Thanos Koutroubas, Afroditi Blika, Anastasis Berdelis, Evangelos Karakolis, Christos Ntanos, Evangelos Spiliotis, and Dimitris Askounis. Adversarial machine learning: A review of methods, tools, and critical industry sectors. *Artificial Intelligence Review*, 2025. Discusses comprehensive taxonomies and robustness benchmarks.

[34] Nikolaos Pitropakis, Emmanouil Panaousis, Thanassis Giannetsos, Eleftherios Anastasiadis, and George Loukas. A taxonomy and survey of attacks against machine learning. *Computer Science Review*, 34:100199, 2019.

[35] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[36] Andreas M. Oberländer, Björn Lösser, and Dennis Rau. Taxonomy research in information systems: A systematic assessment. In *Proceedings of the European Conference on Information Systems (ECIS)*, 2019.

[37] OWASP. OWASP top 10 for large language model applications, 2025.

[38] Cloud Security Alliance. CSA large language model threats taxonomy, 2024.

[39] Cisco. Ai security and safety framework, 2025.

[40] FS-ISAC. Adversarial AI frameworks: Taxonomy, threat landscape, and control frameworks, 2024.

[41] Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, Zhixing Tan, Junwu Xiong, Xinyu Kong, Zujie Wen, Ke Xu, and Qi Li. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. *arXiv preprint arXiv:2401.05778*, 2024.

[42] S. Wang, Tianqing Zhu, Bo Liu, Ming Ding, Dayong Ye, Wanlei Zhou, and Philip S. Yu. Unique security and privacy threats of large language models: A comprehensive survey. *arXiv preprint arXiv:2406.07973*, 2024.

[43] 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.

[44] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, 4(2):100211, 2024.

[45] B. C. Das, M. H. Amini, and Y. Wu. Security and privacy challenges of large language models. *arXiv preprint arXiv:2402.00888*, 2024.

[46] M. Q. Li and B. C. M. Fung. Security concerns for large language models: A survey. *arXiv preprint arXiv:2505.18889*, 2025.

[47] E. Derner, K. Batistič, J. Zahálka, and R. Babuška. A security risk taxonomy for prompt-based interaction with large language models. *arXiv preprint arXiv:2311.11415*, 2023.

[48] S. Rossi, A. M. Michel, R. R. Mukkamala, and J. B. Thatcher. An early categorization of prompt injection attacks on large language models. *arXiv preprint arXiv:2402.00898*, 2024.

[49] Apurv Verma, Satyapriya Krishna, Sebastian Gehrmann, Madhavan Seshadri, Anu Pradhan, Tom Ault, Leslie Barrett, David Rabinowitz, John Doucette, and NhatHai Phan. Operationalizing a threat model for red-teaming large language models. *arXiv preprint arXiv:2407.14937*, 2024.

[50] Arijit Ghosh Chowdhury, Md Mofijul Islam, Vaibhav Kumar, Faysal Hossain Shezan, Vinija Jain, and Aman Chadha. Breaking down the defenses: A comparative survey of attacks on large language models. *arXiv preprint arXiv:2403.04786*, 2024.

[51] A. Esmradi, D. W. Yip, and C. F. Chan. A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models. *arXiv preprint arXiv:2312.10982*, 2023.

[52] Nicholas Jones, Md Whaiduzzaman, Tony Jan, Amr Adel, Ammar Alazab, and Afnan Alkreisat. A CIA triad-based taxonomy of prompt attacks on large language models. *Future Internet*, 17(3):113, 2025.

[53] Farzana Zahid, Anjalika Sewwandi, Lee Brandon, Vimal Kumar, and Roopak Sinha. Securing educational LLMs: A generalised taxonomy of attacks and DREAD risk assessment. *arXiv preprint arXiv:2508.08629*, 2025.

[54] Carlos Peláez-González, Andrés Herrera-Poyatos, Cristina Zuheros, David Herrera-Poyatos, Virilo Tejedor, and Francisco Herrera. A domain-based taxonomy of jailbreak vulnerabilities in large language models. *arXiv preprint arXiv:2504.04976*, 2025.

[55] Muhammad Usman, Ricardo Britto, Jrgen Brstler, and Emilia Mendes. Taxonomies in software engineering. *Inf. Softw. Technol.*, 85(C):43–59, May 2017.

[56] B. Omair and A. Alturki. An improved method for taxonomy development in information systems. *International Journal of Advanced Computer Science and Applications*, 11(4), 2020.

[57] A. M. Oberländer, B. Lösser, and D. Rau. Taxonomy research in information systems: A systematic assessment. In *Proceedings of the 27th European Conference on Information Systems (ECIS 2019)*, Stockholm and Uppsala, Sweden, 2019.

[58] Yannis Katsis, Sara Rosenthal, Kshitij Fadnis, Chulaka Gunasekara, Young-Suk Lee, Lucian

Popa, Vraj Shah, Huaiyu Zhu, Danish Contractor, and Marina Danilevsky. mtrag: A multi-turn conversational benchmark for evaluating retrieval-augmented generation systems, 07 2025.

[59] Zhiyu Chen, Biancen Xie, Sidarth Srinivasan, Manikandarajan Ramanathan, Rajashekar Maragoud, and Qun Liu. LLM-based dialogue labeling for multiturn adaptive RAG. In Saloni Potdar, Lina Rojas-Barahona, and Sebastien Montella, editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1044–1056, Suzhou (China), November 2025. Association for Computational Linguistics.

[60] Ziqi Zhu, Tao Hu, Honglong Zhang, Dan Yang, HanGeng Chen, Mengran Zhang, and Xilun Chen. Conversational intent-driven graphrag: Enhancing multi-turn dialogue systems through adaptive dual-retrieval of flow patterns and context semantics, 06 2025.

[61] Ziyi Liu. A state-update prompting strategy for efficient and robust multi-turn dialogue, 2025.

[62] Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. 2024.

[63] Haneul Yoo, Yongjin Yang, and Hwaran Lee. Code-switching red-teaming: Llm evaluation for safety and multilingual understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, 2025.

[64] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[65] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017.

[66] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Trans. Mach. Learn. Res.*, 2023, 2023.

[67] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in nlp. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

[68] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *arXiv preprint arXiv:2307.02483*, 2023.

[69] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, 2022.

[70] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng,

Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[71] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, et al. On the opportunities and risks of foundation models. *Stanford CRFM Report*, 2021.

[72] Paul Röttger, Giuseppe Attanasio, Felix Friedrich, Janis Goldzycher, Alicia Parrish, Rishabh Bhardwaj, Chiara Di Bonaventura, Roman Eng, Gaia El Khoury Geagea, Sujata Goswami, et al. Msts: A multimodal safety test suite for vision-language models. *arXiv preprint arXiv:2501.10057*, 2025.

[73] China Daily. 10 things you should not give as a chinese new year gift. `https://language.chinadaily.com.cn/2018-02/17/content_35678207.htm`, February 2018. Accessed: 11 January 2026.

[74] Mabel Lui. Gifts you must avoid giving in chinese culture, from blocks and shoes to pears – and why. `https://www.scmp.com/lifestyle/chinese-culture/article/3316324/gifts-you-must-avoid-giving-chinese-culture-clocks-and-shoes-pears-and-why`, June 2025. Accessed: 11 January 2026.

[75] L. D. Dan. Interpretation of taboos in giving gifts to chinese people: Semiotics perspective. `https://www.growingscholar.org/journal/index.php/TIJOSSW/article/download/29/26/89`, June 2020. Accessed: 11 January 2026.