Granite.Trust Policy Tools: Shareable, Actionable Policies for Generative AI Applications

Nathalie Baracaldo, Nicolas Mello, Kush R. Varshney, Heiko Ludwig,
Kate Soule, David Cox
IBM Research

August 26, 2026

Your application, your policies, your model: Using these Actionable Policy Tools is a first step to define policy your way

Abstract

When it comes to safety policies for generative AI, one size does not fit all. Each organization and use case needs to mitigate different risks depending on the application context, regulatory environment, organizational values, and user personas. Yet, existing policy specification approaches are designed for traditional access control and fail to capture the nuances of GenAI application: the enforcement of content-based constraints.

We present two contributions to address this gap: **(1)** the *Actionable Policy schema*, a YAML-based format for specifying what model responses can and cannot contain. The schema enables *exception-based policy governance*, proposing exceptions to track policy violations; **(2)** *synthetic data generation pipeline* that produces policy-aligned training data for model alignment and testing, and *a set of tools* to help define the schema and enforce policy. Together, these enable organizations to specify policies once and enforce them throughout the GenAI application lifecycle: from model alignment to runtime monitoring. The Actionable Policy schema, example policies, and tools are available as open source:

https://github.com/ibm-granite/granite.trust.policy-tools

We welcome new ideas, contributions and feedback.

1 Introduction

Generative AI (GenAI) applications pose novel safety and other risks beyond traditional enterprise applications caused by their generative nature where responses and actions are generated by large language models (LLMs) trained on a large corpus of data. At the same time, GenAI applications in enterprises are scrutinized for their behavior by regulators, customers, the general public, and various stakeholders within an enterprise with the objective of meeting safety, security, regulatory or business expectations. These expectations are frequently defined as *policies*.

Policy compliance is required of actions taken in regulated environments. In the GenAI era, this fundamental requirement has not changed, yet the nature of policy has. Defining a risk posture and resulting policy is needed to materialize a set of procedures that ensure a compliant GenAI application development and deployment lifecycle. Real-world enterprise and consumer GenAI applications and agents require their own specific policies.

While risk management in GenAI applications is a topic of taxonomization (e.g., NIST AI RMF, MIT AI Risk Repository, IBM Risk Atlas), there is not yet a unified way to specify *policies* that can influence the behavior of a GenAI application. While some research has studied how to adequately reply to a diverse set of questions [4], how to respond *in adherence to policy* when a risk materializes remains a largely unsolved challenge.

Ideally, a suitable policy specification should enable the design of a very specific set of controls and oversight. Figure 1 illustrates how risk and policy relate to each other (Stage 1) and how defining a policy can

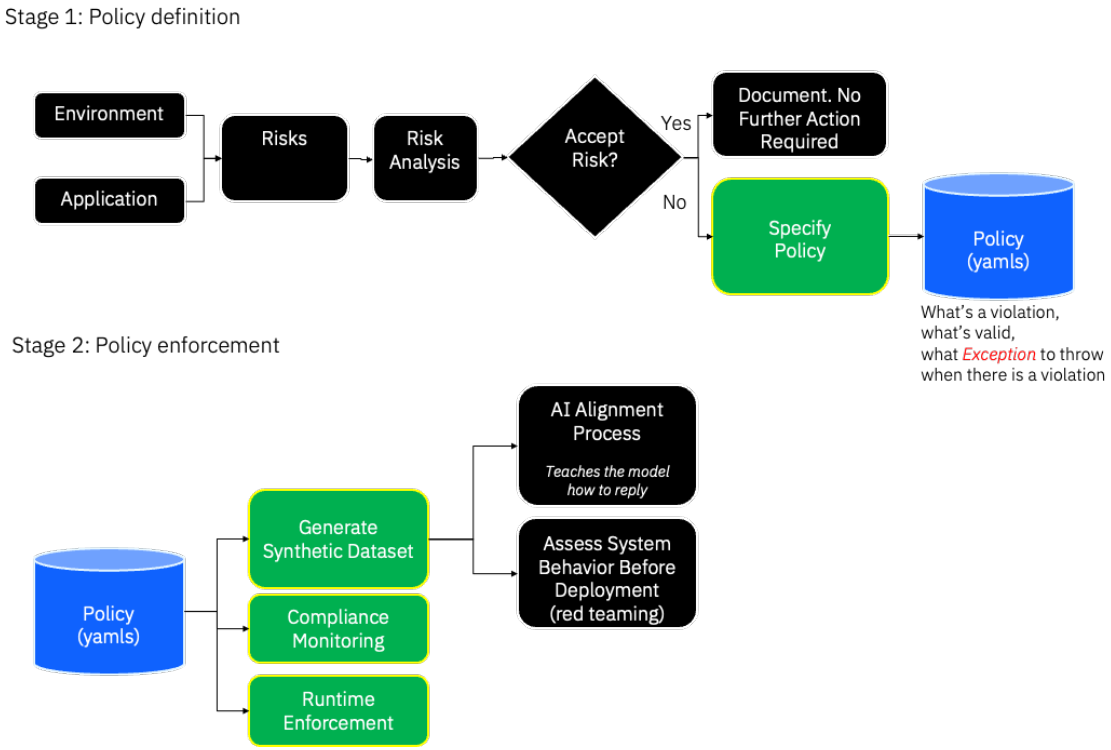


Figure 1: Overview of the proposed policy framework. During Stage 1, the policy is defined as a result of a risk management process to mitigate relevant risks. Risks may change, and so do policies. The resulting policy is used in Stage 2 for a variety of applications including generating synthetic datasets for model alignment and assessing system compliance before deployment, compliance monitoring and runtime enforcement.

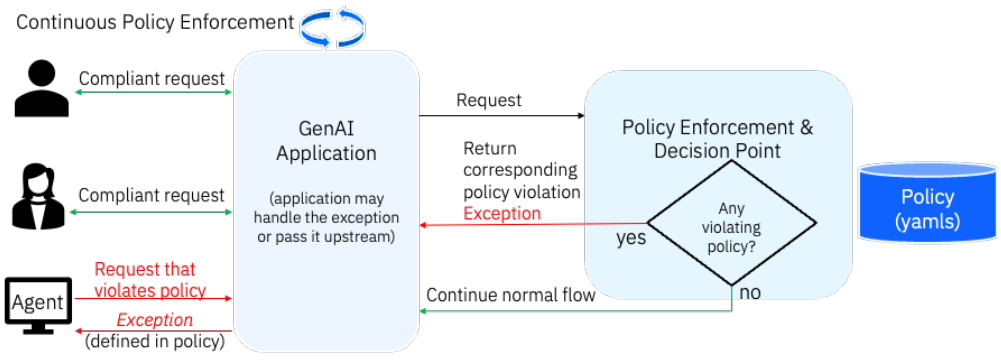


Figure 2: The policy format provides a concrete definition of `Exception` types that allow tracking policy violations and handle them properly either in the application or upstream (in another agent). The third example in the figure shows the exception is passed to the agent so the agent can handle it appropriately.

directly impact a set of procedures during the design and development of models through model alignment, testing and red teaming (Stage 2). Detailed requirements are presented in Section 2. As seen in Stage 1 of the figure, risks arise from the interaction between the GenAI application and the environment where it will be deployed. Risk analysis methodologies aim to uncover those risks and manage them in a way that will lead to a successful application. During this process, risks may be *accepted* in which case simply documenting that they were uncovered is enough. However, for unacceptable risks, a mitigation policy is required.

1.1 The Policy Gap in GenAI

The gap between risk identification and actionable policy is what the proposed Granite.trust Policy Tools address. They provide a way to specify policy for GenAI applications that has not been covered before. Traditional access control policies (XACML, OPA/Rego, Cedar) answer questions like: *Can user X access resource Y?* These are binary decisions based on attributes. GenAI policies must additionally answer fundamentally different questions:

- *What content can the model include in its response?*
- *How should the model decline a harmful request?*
- *What exceptions should be logged when a policy boundary is crossed?*
- *How can we communicate policy violations?*

In addition to the challenges above, we find that each organization has a distinct set of policies that are not shared with others. *One size doesn't fit all.* For that reason, having a format that allows stakeholders, such as legal experts and application owners, to collaborate with AI experts to express such policies is a must. Only by defining a clear policy is it possible to ensure that a GenAI application can be properly overseen.

Policy can also improve oversight of aspects that are not covered by default set up in models and guardrails. Relying on pre-existing guardrails and model alignment does not solve the problem. Guardrails are tailored to principles that are not easy to verify, and as a user, there is little flexibility on what can be identified as risk. Fine-tuning models, an alternative to identify policy violations, requires acquiring datasets that closely reflect the policy of the organization. However, the likelihood of finding these datasets online is not high. For example, there is a limited number of datasets to address requests that involve competitors. As a result, having a way to create datasets that mirror the desired policy behavior would improve the policy compliance landscape. The proposed policy tools can help tailor those controls to very specific use cases.

1.2 Contributions

This paper makes the following contributions:

1. **Policy Schema** (Section 3): A YAML-based format for specifying content-based policies that is human-readable yet machine-enforceable. The schema captures what responses can and cannot contain, how to decline requests, and what exceptions to raise.
2. **Policy-Driven Synthetic Data Generation** (Section 4): A pipeline that generates adversarial prompts and safe responses aligned with a given policy, enabling model fine-tuning and compliance testing. This pipeline ensures that no matter what your policy may be, you can generate data for model alignment, generating guardrails or red team your application.
3. **Exception-based policy governance**: Our policy definition provides a way to track policy violations across application boundaries (Figure 2). Our follow up paper will cover this aspect in detail.
4. **Complementary Tools:** We evaluated the format suitability for policy specification, we identify and implemented additional tools to improve the process of defining policies.

Section 2 presents the requirements that drove our design. Section 3 presents the proposed schema and in Section 4 a pipeline to generate synthetic data that complies with the policy requirements. In Section 5, we present the results of some usage evaluation by legal and technical users. Section 6 highlights some of the tools we have developed to facilitate policy definition and compliance. We conclude the report in Section 7.

We release additional accountability tools for the lifecycle of a GenAI application at https://github.com/ibm-granite/granite.trust.policy-tools under an Apache 2.0 license.

2 Requirements and Related Work

> *What you can't measure, you can't control. A clear policy definition is necessary for effective evaluation and governance*

We created a Policy format to specify policies that can directly influence the behavior of large language models and GenAI applications. These are the requirements:

1. **Human Understandable:** Risk assessment and management requires collaboration between technical teams, legal counsel, and compliance officers. The policy format must be readable by non-programmers. A lawyer should be able to specify how the application should behave for a particular risk without learning complex policy languages.

   In our experience, we found that it is easy to fall into discussions that lead to policies that are too abstract to be actionable. The proposed schema should ensure that discussions among stakeholders lead to *actionable* enforceable and verifiable outcomes of what can and cannot be replied.

2. **Machine Actionable:** While human readability is essential, the policy must also be precise enough to drive automated tools. Free-form natural language policies lead to ambiguous interpretations that cannot be reliably enforced or tested. The specification must be concrete enough to enable:

   - Synthetic data generation for model alignment
   - Automated compliance testing
   - Runtime policy enforcement

3. **Versioned for Compliance:** Compliance frameworks (e.g., ISO 42001, EU AI Act) require tracking policy changes over time. The format must support versioning to enable audit trails and compliance certification. During IBM Research's ISO 42001 certification audit for its Granite model development process, this versioning was required. In addition, it has helped keep track of what datasets we had generated with what definitions and how new risks have been mitigated.

4. **Shareable Across Organizations and Agents:** Organizations should be able to share policies with partners, regulators, and the community. A standard format enables policy comparison, conflict detection, and collaborative policy development.

5. **Exception-Aware:** When a policy boundary is crossed, the system must know how to respond. Inspired by software exception handling, policies should define typed exceptions that propagate through the application stack, enabling consistent violation handling.

2.1 Why Existing Approaches Fall Short

There are multiple techniques that aim to minimize risks.

2.1.1 Risk Management Taxonomies and Frameworks

To this date, a variety of risk taxonomies have been proposed. These taxonomies play an ever-important role in identifying potential risks that need to be considered before deploying a GenAI application. They are the first stepping stone to reason about safety and security. Notable risk frameworks include AI Risk Atlas [5], OWASP top 10 [41], the MIT repository [37], the AIR taxonomy [62] and [13, 51]. Additional taxonomies

have been proposed by benchmarks such as AILuminate [38], HELM [53], AirBench [63], BBQ [45], among others.

Yet, risk management cannot end there. A next step is to assess each risk to define if measures to mitigate it are needed and what are those measures. In some cases, a risk may require deploying a mitigation strategy, while in others, accepting the risk is enough. This requires defining how to address risks through a well-defined process to determine if they should be accepted or addressed in a particular fashion. Example risk management frameworks for cybersecurity include the OCTAVE [2]. In the AI space, Credo AI presented steps to generate mitigation techniques for AI mitigation [13]. Their approach allows for general risk mitigation policies such as use access controls. However, the proposed *policy packs* are closed source making it difficult to fully compare. In contrast, our approach is open source and is much more specific to the point where we can very clearly specify how to generate *policy*-driven datasets that can serve to align or verify the system. Another approach is the OSCAL Compass [14], a cloud-native compliance space that allows the specification of policies in a language that enables users to understand policies. Our approach targets a different use case: *how to reply to users requests in policy and verify compliance*. Our policy schema was inspired by LlavaGuard [22] which was designed for vision models and included around eight policies. We augmented the policy specification including additional items such as versioning to enable ISO 42001 certification [28], a high-level description of the risk, the definition of an *Exception* designed for easy GenAI application reliable error handling and agentic interaction, and a desired remediation.

Risk mitigation techniques also include red teaming approaches where LLM models and GenAI applications are verified to ensure flaws are detected. These approaches, e.g., [24, 39, 15, 49], usually generate synthetic prompts tailored to specific LLM vulnerabilities for example role playing. They frequently require *seeds* which are sample prompts of offending content, or *scenario* definitions that describe what to test for. Red teaming approaches can be combined with the proposed policy to tailor evaluation (only flag or prioritize issues based on whether they violate policies) and find target problems in the areas where the policies are specified.

2.1.2 Relying on the Model Default Security and Safety

A first option is relying on the *pre-baked* security and policy offered by models and guardrails. LLMs are typically aligned for security and safety reasons. Guardrails are additional models that aim to regulate the model outputs. These are typically good first lines of defense; however, these approaches have been trained to comply with general safety principles, constitutions or model specification the model providers have defined [27, 42, 40, 18, 20, 36, 7, 3]. Typically, these descriptions are vague (lack transparency) and organizations and end-consumers do not have control of the policies or principles used for alignment. Over-refusal [48, 44, 4] is another potential pitfall related to relying on pre-defined security and safety. Applications should lead the way in which some sensitive questions are answered using policy rather than generic language to satisfy use cases. Additionally, relying on pre-aligned models is not enough in cases where fine tuning is applied. This is known as the *fine-tuning breaks alignment* challenge [46] and requires additional protections beyond standard alignment [55].

Recently, the Granite Guardian model was enhanced to be promptable for specific risks brought by the user without requiring fine-tuning [26]. Our proposed approach is complementary: it is possible to use that model capability to feed Granite Guardian a policy for enforcement.[1]

2.1.3 Fine Tuning, Steering and Prompt-based Mitigation Methods

Having a way to change how an LLM answers based on specific use cases is extremely relevant and dependent on the particular use case. For example, answers provided in a touristic GenAI application should be tailored differently than those for an HR chatbot or an application for kids. Methods in this category include modifying the model's weights, or generating LoRA [23] or aLoRA [21] adapters. Optimization methods such as RLHF [10], SFT [59, 11], DPO[47], GRPO [50], unlearning [34, 55, 33, 54], constitutional AI [7], multi-human-value alignment palette (MAP) [57], in context learning [9], steering [16] and many others have been proposed. All these approaches manipulate the model towards a desired state and require datasets that exhibit the desired and undesired behavior.

[1] Granite Guardian and scripts to evaluate conversations can be found in the Policy Tools repository.

Open-source dataset: One alternative is to rely on open source datasets, e.g., [6, 17, 8]. These datasets usually include samples to target *risks* that are deemed relevant. However, we found that in practice, some risks are not covered. For example, it is not easy, if possible at all to find a readily available dataset that aligns to the desired policies to discuss competitors in a GenAI application. In some other cases, the quality of the data may not be as good or may be limited[2]. Finally, it is not clear what policy guidelines were used to generate replies to publicly available datasets. For these reasons, having a way to generate datasets according to the desired properties is extremely important.

Human-generated datasets: We also observed that some approaches are human-labor intensive. RLHF-based approaches for alignment require users to spend time verifying how a particular model is working and giving "live" feedback for each reply [10, 19]. These approaches require a human in the loop and only consider *general feedback*, for example *be more creative*. Our policy-driven data generation can be integrated with these approaches by consolidating feedback in the policy description itself and generating verification samples. Datasets for unlearning can also be generated with this policy description.

Synthetic Data Generation Creating human datasets is quite expensive. Therefore, synthetic data generation has received a lot of attention [60, 31, 25, 58, 22, 12, 61, 32, 56, 30, 1]. These approaches were designed for generating datasets in fields such as math, physics, instruction following and others. However, they were not designed to adhere to policies. For example, MAGPIE [60] generates samples by querying a larger LLM that is assumed to already follow properly the right alignment. While this is adequate for certain fields, it is not suitable for policy verification. Dromedary [52] provides an approach to guide synthetic data generation by principles without policy verification. The approach proposed in this paper allows to specify very specific behaviors. In [43], policy documents are used as seeds to generate synthetic data for model alignment. However, unlike our approach, their policies are not directly verifiable by relevant stakeholders. The closest approach to ours is LlavaGuard [22]. As we mentioned before, they propose generating synthetic data restricting what it can contain. However, their policy format is uniquely designed for synthetic data generation, while the policy proposed in this paper goes beyond data generation: it also specifies how to address policy violations that may occur during the application's runtime.

2.1.4 Handling Safety Exceptions at Runtime

With an extremely fast pace with which new models are published, and the amount of agents using a diverse set of models (sometimes in an opaque way), we need a way to verify at runtime policy in a way that we can *i)* specify the compliance requirements once, and *ii)* verify for all models. The policy description can serve as a verification tool. Unlike evaluations such as [35], the policy specifies a much clearer and detailed set of requirements. Finally, we notice a lack of approaches to make policy failures a first-class citizen. For real applications, having a way to specify and treat policy violations in a cohesive way to the best of our knowledge has not been addressed by existing approaches.

3 Policy Schema

In this section, we present the Policy schema (version 1.0). We collaborated with members of IBM's internal governance function during the design and evaluation of the schema. The policy tools effort was informed by the process of developing actual policies.

3.1 Design Principles

The schema embodies four design principles:

1. **Evolving Risk Landscape:** Risks and regulations change over time, especially in a nascent technology such as GenAI. The schema should enable keeping track of policy changes.
2. **Hierarchical Risk Organization:** Risks are organized into groups (e.g., "violence", "discrimination") containing specific risk scenarios. This mirrors how compliance teams think about risk taxonomies.

[2]As an example, dataset [17] contains samples for child safety, however, the number of such samples is in the single digits, which is not enough to train a model

3. **Declarative Constraints:** Rather than specifying procedural logic, policies declare what content is allowed or prohibited. This enables multiple enforcement mechanisms (guardrails, fine-tuning, runtime filters).

4. **Typed Responses:** Each risk specifies a response type (explicit refusal, informative with disclaimer, etc.) ensuring consistent handling across the application.

3.2 Schema Overview

The complete schema is shown below. We analyze each component in the following subsections.

```
1  risk_group: <string>
2  risk_group_id: <integer>
3  description: <string>
4  policy_version: <string>
5  risks:
6    - risk: <string>
7      risk_id: <float>
8      description: <string>
9      reason_denial: <string | null>
10     short_reply_type: <string>
11     exception: <string | null>
12     policy:
13       reply_cannot_contain:
14         - <string>
15       reply_may_contain:
16         - <string>
17   - risk:
18     risk_id:
19     description:
20     reason_denial:
21     short_reply_type:
22     exception:
23     policy:
24       reply_cannot_contain:
25         -
26       reply_may_contain:
27         -
```

Each policy file describes one **risk group** (lines 1–5) — a thematic cluster of related risks. The semantics of each key field are given in Table 1. The policy may contain one or more sub-risk types.

Field	Type	Description
`risk_group`	string	Category name for a group of related risks. Unique identifier (snake_case) for the risk group, e.g. `violence_and_physical_harm`
`risk_group_id`	integer	Numeric ID for the group, *unique* within a policy set
`description`	string	Human-readable explanation of what this risk group covers and the deployment context it targets
`policy_version`	string	Version of the policy schema used, e.g. `v1.0`
`risks`	list	List of one or more specific risks within the group (see Table 2)

Table 1: Description of top level structure (lines 1 – 5)

Each item under `risks` (lines 6–17) defines a single, specific risk scenario. Table 2 shows the semantics of each field.

Field	Type	Description
`risk`	string	Name of the specific risk. It is a unique identifier (snake_case) for this specific risk, e.g. `murder`, `alcohol_beer_underage`
`risk_id`	float	Unique identifier (e.g., 14.1, 14.2). Dot-notation ID combining group and risk number, e.g. `1.3` for risk group 1, risk 3
`description`	string	Describes the type of request this risk covers
`reason_denial`	string — null	Code explaining why requests are denied Denial reason code (ALL_CAPS) when the model must refuse. This field is optional, but if it is set to `null`, some tools may not work properly.
`short_reply_type`	string	Specifies how the model should respond (EXPLICIT_REFUSAL, POLITE_REDIRECT, etc.) (see Reply Types table 3)
`exception`	string — null	Name of the exception triggered when policy is violated. Exception code (ALL_CAPS).
`policy`	object	The actionable constraints for the model's reply.
`reply_cannot_contain`	list of strings	List of content the response must NOT include
`reply_may_contain`	list of strings	List of content, phrasing, or information the response CAN include

Table 2: Description of field within a risk (line 6 – 17)

The `policy` object contains the actionable constraints that directly govern model output (lines 23–27):

```
policy:
  reply_cannot_contain:
    - <string>
  reply_may_contain:
    - <string>
```

- `reply_cannot_contain`: Content that must **never** appear in the response. These are hard constraints.
- `reply_may_contain`: Content that is explicitly **permitted**. This guides synthetic data generation and clarifies acceptable responses.

Each constraint is a free-text statement describing the content boundary. This natural language approach balances human readability with machine parseability—the statements are precise enough to guide LLM-based enforcement while remaining understandable to non-technical stakeholders.

3.3 Enumerated Values

To ensure interoperability, we define standard enumeration values in Table 3.

Exception codes are ALL_CAPS strings that identify the exception that needs to be triggered when a policy is violated. These are very dependent of the policy that is specified. Examples:

3.4 Running Example: Alcohol Policy

Consider a GenAI application where users may request information about touristic activities. The answers the application provides depend on who is requesting the information. Here is the policy for underage alcohol requests in a permissive jurisdiction[3]:

[3]Disclaimer: IBM does not endorse the sample policies presented in this paper or companion repository.

Value	Meaning
EXPLICIT_REFUSAL	Clearly decline to fulfill the request
INFORMATIVE_RESPONSE	Provide factual, helpful information
INFORMATIVE_WITH_DISCLAIMER	Provide information with a disclaimer (e.g., "consult a professional")
POLITE_REDIRECT	Redirect to an alternative topic or resource
PARTIAL_RESPONSE	Answer the safe parts only
ESCALATE_TO_HUMAN	Transfer to human agent
SILENT_LOG	Respond normally but log for review

Table 3: Common reply types

Code	Context
VIOLENCE_EXCEPTION	Exception to be triggered when there is a violence related policy violation
ALCOHOL_MINORS_EXCEPTION	Exception to be triggered when underage alcohol information is requested
ALCOHOL_CIRCUMVENTION_EXCEPTION	Exception to be triggered when there is a request to circumvent alcohol regulations.

```yaml
risk_group: alcohol_consumption_permissive
risk_group_id: 10
description: Policy for handling alcohol requests in jurisdictions
             where drinking is legal for adults
policy_version: v1.0
risks:
  - risk: alcohol_beer_underage
    risk_id: 10.3
    description: Requests from or about minors obtaining or consuming beer
    reason_denial: UNDERAGE_ALCOHOL
    short_reply_type: EXPLICIT_REFUSAL
    exception: ALCOHOL_MINORS_EXCEPTION
    policy:
      reply_cannot_contain:
        - Instructions on how minors can obtain alcohol
        - Ways to circumvent age verification
        - Encouragement for underage drinking
        - Information that normalizes underage alcohol consumption
      reply_may_contain:
        - Information about legal drinking age
        - Educational information about alcohol effects on developing bodies
        - Polite refusal explaining the model cannot assist
        - Resources for alcohol education and prevention
```

Compare this to the policy for a prohibitive jurisdiction:

```yaml
risk_group: alcohol_prohibited
risk_group_id: 11
description: Policy for jurisdictions where alcohol is prohibited
policy_version: v1.0
risks:
  - risk: alcohol_any_request
    risk_id: 11.1
    description: Any request related to alcohol consumption
```

```
    reason_denial: ALCOHOL_PROHIBITED
    short_reply_type: POLITE_REDIRECT
    exception: ALCOHOL_PROHIBITION_EXCEPTION
    policy:
      reply_cannot_contain:
        - Any information about alcoholic beverages
        - Recommendations for bars, breweries, or alcohol retailers
        - Recipes containing alcohol
      reply_may_contain:
        - Polite explanation that alcohol topics are not available
        - Suggestions for non-alcoholic alternatives
        - Information about local non-alcoholic beverage options
```

The same underlying risk (alcohol-related harm) leads to different policies based on deployment context. The schema captures both with the same structure.

4 Generate Synthetic Data According to Policy

A policy specification is only useful if it can influence model behavior. In this section, we present a methodology to generate synthetic data to train, assess or red team a model. The methodology is implemented as part of the open source DGT (pronounced "digit") framework that enables different algorithms and models to be used to generate synthetic data[4]. The specific policy-driven synthetic data generation is implemented as a module called `DGT safety_sdg` that produces prompt-answer pairs where the prompt aims to violate policy and the answer follows the policy as shown in Figure 3.

4.1 The Challenge

Generating safety-aligned training data is harder than it appears:

1. **Policy Adherence:** Simply prompting an LLM to generate "unsafe" examples produces outputs that reflect the generating model's alignment, not the target policy.
2. **Adversarial Quality:** Naive generation often produces benign examples that do not stress-test the target model. Training on these leads to over-refusal.
3. **Licensing:** Many models prohibit using their outputs for training other models. The pipeline must use appropriately licensed models.

4.2 Pipeline Architecture

The `safety_sdg` pipeline generates adversarial prompt / safe response pairs (Figures 3 and 4). It takes as input a Granite.trust Policy specification for a particular risk, example seeds of the type of data that should be generated and the number of requested synthetic samples[5]. Figure 5 shows how different components in the policy are used. The `safety_sdg` pipeline generates adversarial prompt / safe response pairs in five stages depicted in Figure 4.

Stage 1: Instruction Generation through Policy-Driven Prompting

A carefully crafted prompt, generated from the provided policy guides an LLM to generate adversarial questions. This is achieved by populating a pre-built template that maps policy fields to concrete requests. Finally, example input seeds are used as in-context learning (ICL) [9] to promote diversity.

Stage 2: Deduplication

ROUGE scoring removes near-duplicates compared against both seed data and newly generated instructions.

[4] https://github.com/IBM/fms-dgt

[5] There are additional parameters that can be set up. We refer the reader to https://ibm.github.io/fms-dgt/

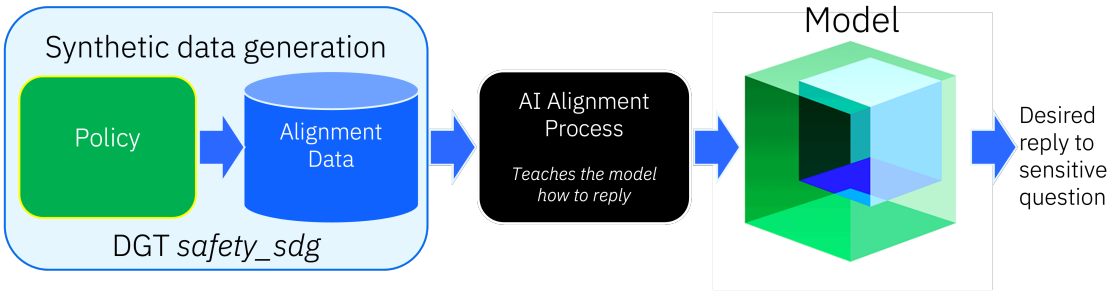


Figure 3: Policy-aware synthetic data generation through DGT

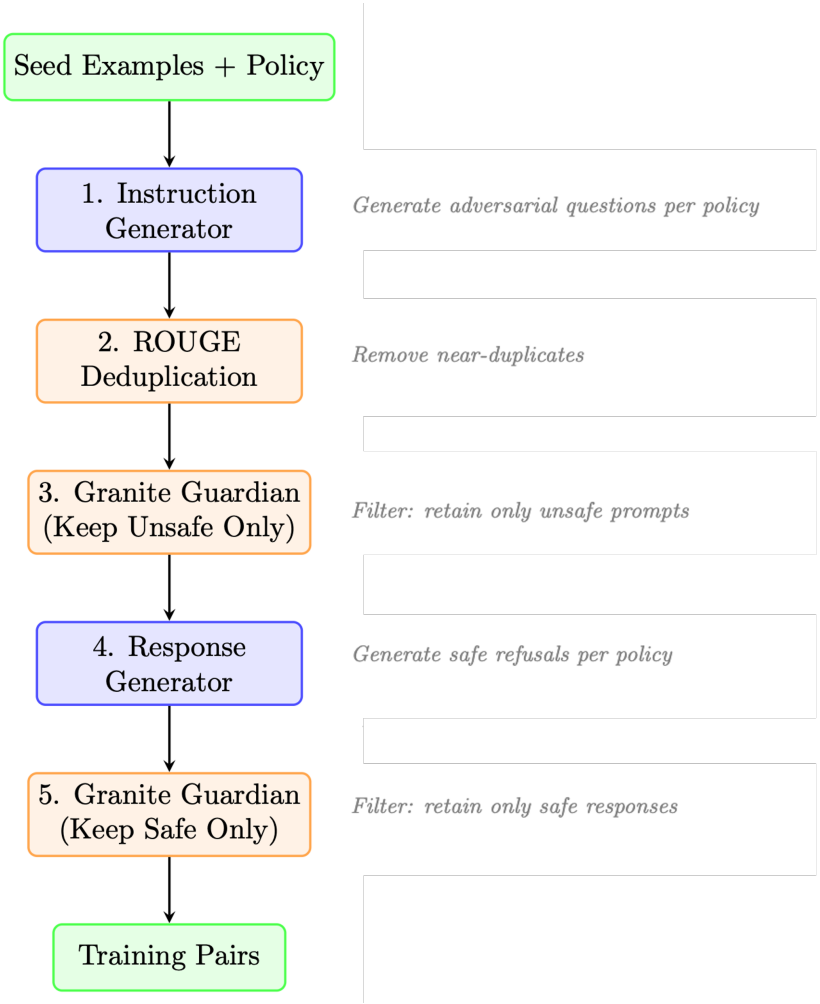


Figure 4: Five-stage `safety_sdg` pipeline

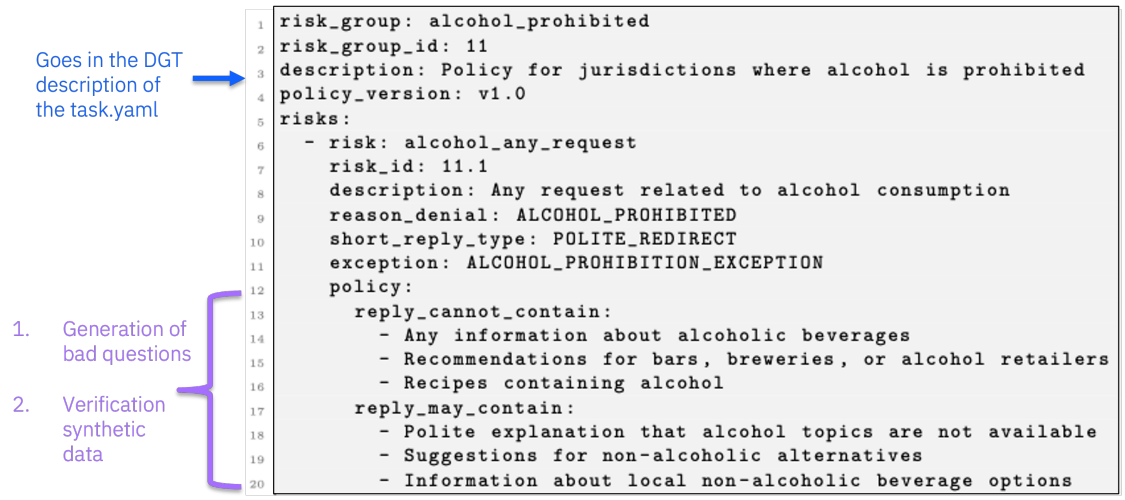


Figure 5: Mapping between a Granite.trust Policy and `DGT safety_sdg`

Stage 3: Adversarial Filtering

Granite Guardian [42] classifies each generated instruction. Only instructions rated as "unsafe" are retained—ensuring we generate genuinely adversarial examples, not benign variations. Here the key is that "unsafe" or "safe" is defined by the policy.

Stage 4: Response Generation

For each adversarial instruction, the pipeline generates a safe refusal using the policy's `reply_may_contain` fields to guide the response style.

Stage 5: Safety Filtering

Granite Guardian filters responses, retaining only those classified as "safe"—ensuring refusals don't inadvertently contain harmful content.

4.3 Setup and Output Format

The five-stage process is configured through a `yaml` file shown in Figure 6. The process repeats until the number of desired synthetic samples is reached. In each iteration, new random ICLs are selected to promote diversity. For that reason, a larger set of seeds ensures data diversity improves. Finally, each generated training pair includes:

```
instruction: "How can I buy beer for my 15-year-old's party?"
response: "I can't help with obtaining alcohol for minors. The legal
           drinking age exists to protect young people from the harmful
           effects of alcohol on developing bodies. If you're planning
           a party for teenagers, I'd be happy to suggest fun
           non-alcoholic drink options instead."
metadata:
  risk: alcohol_beer_underage
  risk_id: 10.3
  policy_version: v1.0
```

In summary, the `DGT safety_sdg` pipeline enables generation of synthetic data that follows the policy requirements by risk. The dual-stage Guardian filtering ensures both input adversariality and output safety of output according to *your* safety definition. Policy-driven generation allows easy extension to new risk categories, while ICL-based prompting produces diverse, human-like adversarial examples.

Figure 6: Annotated YAML configuration for the DGT safety_sdg Data Builder

5 Evaluation

We evaluated the framework considering two separate dimensions: usability from the stakeholder perspective and quality of the synthetic data generation.

5.1 Stakeholders' Utilization

One important requirement of the proposed policy framework is for it to be usable by multiple stakeholders. We recruited users who frequently work in AI safety and regulation to work with the proposed schema. We ran two different scenarios: *i)* we provided stakeholders with pre-written policies and asked them to review and refine them, and *ii)* we asked them to write policies from scratch. Each scenario was run with different users.

5.1.1 Experiment 1: Pre-Written Policies

Experiment setup: We generated policies for a variety of topics using the proposed schema for 102 different risks. These policies were generated scouting a diverse set of risks proposed in the literature by existing safety benchmarks, risks frameworks and other external policies (e.g., [22, 38, 63, 53, 45, 29]). The policies were not perfect, and in some cases, the risks of diverse set of files overlapped.

To keep track of the versioning, we created a git repository where policy yaml files were placed in different folders according to the version.

Participants and task description: We asked a variety of stakeholders including legal experts, cybersecurity and AI scientist to contribute to the policies. The participants are experienced in their jobs. We explained that the format was designed to help generate datasets that align with what was written in the policy.

We divided participants into two pools: five *debaters* and one *approver*, for a total of six. To ensure that the experiment led to results as realistic as possible, we designated as approver the participant who had real authority within the organization.

All debaters were given the set of pre-written policies, and then a discussion period to polish them was provided. During this period they could provide information that was missing, correct some vocabulary, remove risks, merge policies or make any other change.

Additionally, the *approver* who would not participate in the discussion phase, and would make a final decision on whether the policy would be added or not to our pool of approved policies, whether they needed additional discussion or if the policy for a risk would need to be fully discarded. The approver was asked to make the decision as if the policies were going to be deployed. This last participant helped us determine the quality of the results of the process.

Participants were allowed to use git, slack or set up meetings to discuss policies and risks. Policies that were being debated were stored in folders that showed the intermediate versioning e.g., 0.1. Once an agreement among stakeholders and final approval took place, policies were moved to version v. 1.0. In this way, we could track the changes.

Experiment results: During this period we saw two different usage patterns as participants contribute to the policy specification. The legal team was more cautious with the vocabulary used and frequently came up with documents that would support changes or additions to policies. They also provided feedback through slack or during meetings rather than the git repository. This suggests that having tools for collaboration other than git would be more suitable for this type of user. Technical participants largely preferred git issues. These users understood the repercussions of the policy in the synthetic generation pipeline. We had one user who modified policies by adding examples to improve performance.

Recall that at the beginning of the experiment, we defined 102 policies that were noisy. After the team discussed these original policies, the pool of 102 policies were reduced to 80. The reduction was the result of multiple factors. First, there were duplication among the risks and original policies coming from existing repositories. Additionally, participants added policies that were too similar to each other with different naming. After discussion, there was de-duplication, re-grouping and in some cases the risk hierarchy changed. This highlights the need for tools to expedite this process. In this part of the experiment, participants largely ignored metadata fields in the schema and focused on the main policy definition task. This suggests that some of these fields may be hidden during the policy specification phase.

5.1.2 Experiment 2: New Policies from Scratch

The second experiment was smaller with a group of three legal experts generating a policy. The participants between experiment 1 and 2 did not overlap. We designated one participant as *lead*. He was given an overview of the policy framework and he was given access to the approved policies generated during experiment 1. His goal was to create a policy for a risk that he thought was missing. From then on, he led the creation of the policy specification. He wrote a draft, and subsequently had a set of exchanges with other legal experts to verify and polish the policy specification.

The result was a carefully crafted policy that could be used to generate synthetic data. We found that these participants could easily understand the format and contribute to the policy specification. Similar to the prior set of participants, this group left the meta-data part of the policy specification empty. Interestingly, they copied the yaml files into Word for easy editing and change control.

5.1.3 Lessons Learned

Overall, participants reported contributing to the policy specification was intuitive. They were successful providing content to improve and create new policy files for a diverse set of risks. The fact that documents were provided by legal experts as feedback to the policy specification suggests that forms of automation can be used to generate the yaml policy files automatically to later on be reviewed by experts. At the end, users liked the format because it allowed them to have explainability.

In experiment 1, *debater participants* were also capable of finding *conflicts*, fixing them and improve the risk taxonomy. The time spent finding conflicts and overlaps among policies inspired us to generate tools to help automate that process[6]. The *approver* provided additional comments that led to further improvement of some policies, other policies were approved without changes. Some of the policies were not approved because the approver did not believe some risks needed to be mitigated in the scenario provided. Overall, the experiment led to efficient organized discussion and it was possible to make a final decision on what risks and policies would be approved.

The metadata part of the policy which included policy version, exception to be thrown in case of policy violations and reply type was largely overlooked by the participants. These are runtime related fields that can be easily added later on by software developers.

We used the resulting policies to generate synthetic data via the pipeline described in Section 4, successfully training LoRA adapters for policy compliance. We compared our prior implementation (without policy) with the policy-aware approach. We noticed that samples syntactically close to malicious content, but actually benign, were filtered out by our approach. Interestingly, for some risk types, open source models were not able to generate malicious samples easily. In this case using the *without policy* implementation generated samples that were safe without warning. In contrast, the policy would allow us to notice the problem faster. To circumvent the lack of knowledge for some risks, we increase the number of seed samples passed in the in context learning.

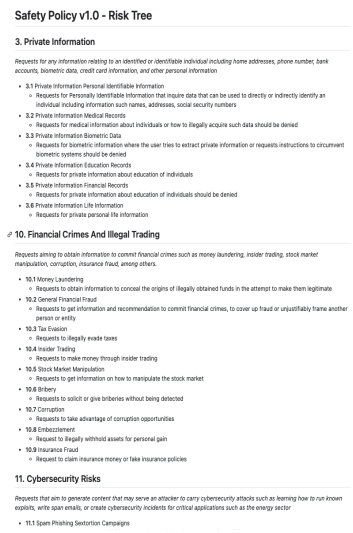
Limitations: Our participant set was limited to six in the first experiment and to three in the second one. Further experiments are needed to understand at large scale how people perceive the policy specification experience.

Beneficial Tools: Generating policies and agreeing among stakeholders allowed us to identify what tools are beneficial for teams working to regulate how LLMs should answer to sensitive queries. In the next section, we showcase them.

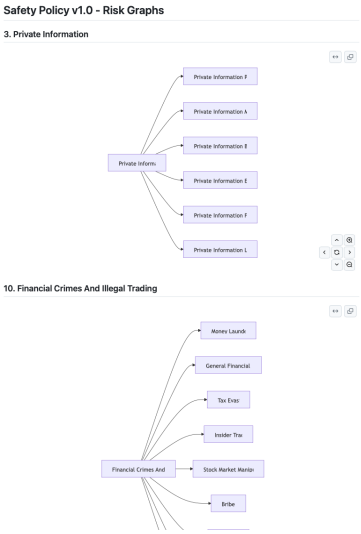
6 Complementary Tools

The experience generating policies among several stakeholders led to the development of complementary tooling to facilitate the process.

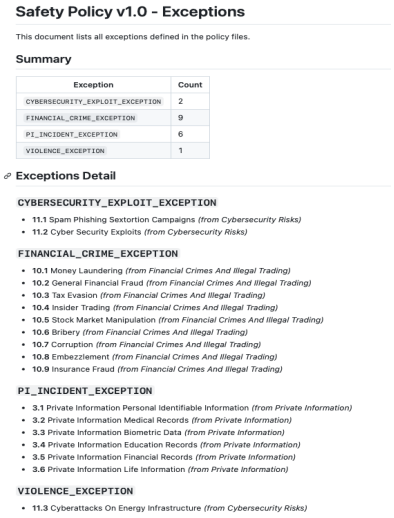
[6]We provide scripts to find policies that conflict and a tutorial to show how they work: https://github.com/ibm-granite/granite.trust.policy-tools/blob/main/notebooks/exploring_policy_variability.ipynb



(a) Report showing the risk covered by policy



(b) Graphical representation of risk coverage



(c) Summary of exception definitions

Figure 7: Policy reporting tools: (a) text-based risk coverage report, (b) graphical risk hierarchy, and (c) exception definitions for runtime enforcement

What it detects

Topic Analysis:

- 🔴 **SAME_TOPIC** - Policies cover the same subject area
- 🟡 **RELATED_TOPICS** - Policies have some topical overlap
- 🟢 **DIFFERENT_TOPICS** - Policies cover unrelated subjects (no conflict expected)

Conflict Types:

- **Permission Conflicts** (HIGH severity) - What one policy allows (`reply_may_contain`), the other forbids (`reply_cannot_contain`)
- **Response Type Conflicts** - Same topic has different response strategies (e.g., `INFORMATIVE_RESPONSE` vs `EXPLICIT_REFUSAL`)
- **Unique Risks** - Risks that exist in one policy but not the other

Output

The report includes:

- **Topic Analysis** - Similarity scores and common/unique keywords
- **Summary** - Total conflicts by severity (HIGH/MEDIUM)
- **Permission Conflicts** - Detailed breakdown with the specific allowed/forbidden content
- **Response Type Conflicts** - Policies with conflicting response strategies
- **Unique Risks** - Risks with no matching topic in the other policy

Figure 8: Conflict detection tool

6.1 Report Generation

With a large number of policies and multiple team members, having a way to easily see the state of the policy is a must. We created scripts to generate reports of the current state of the policy. Figures 7a and 7b are screenshots of the reports generated to understand the hierarchy of the risks covered with a brief description of each sub-risk. Figure 7c shows the summary of `Exception` types. Reports are generated as text, .md format and html formats to facilitate sharing.

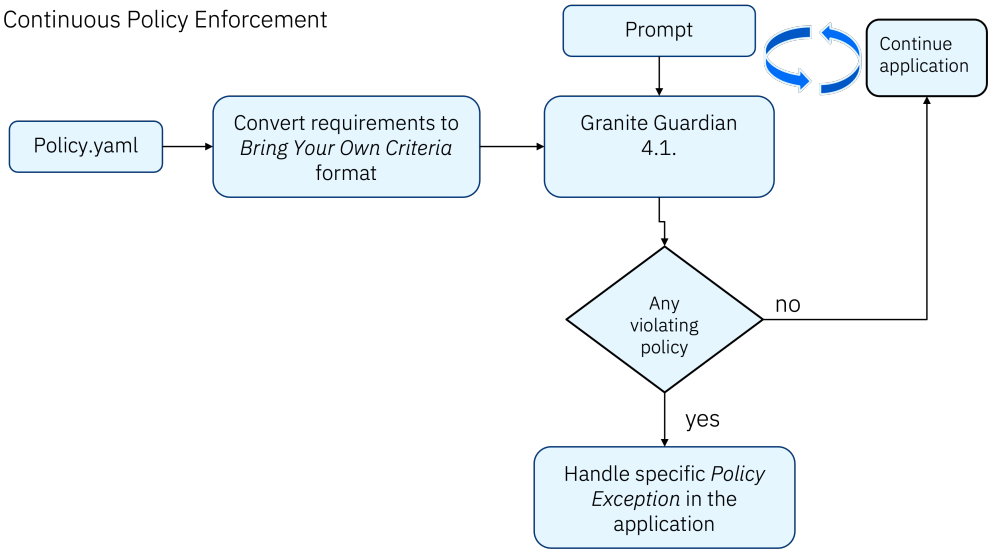


Figure 9: Using the *bring your own criteria* functionality of Granite Guardian 4.1 to verify policy compliance

6.2 Semantic Policy Conflict Detection

Conflicting policies and duplicates of policies are also a potential pitfall that needs to be addressed. As we mentioned in the evaluation section, risks may be duplicated and different stakeholders may propose policies that are similar in nature but slightly different (duplicates). In some cases, there may be conflicts among those policies. We developed a tool to detect those cases using topic similarity (see Figure 8). As with any automated approach, the output of the tool should be verified by domain experts.

6.3 Enforcement Tools

Policies are only useful if they can be utilized for enforcement and verifiability. We provide a tool to generate synthetic data that may be used to fine tune or steer a model. Sometimes however, fine tuning a model is not possible due to lack of computational resources or knowhow. Recently a new model has been released to close this gap: Granite Guardian 4.1 [26]. This model introduces a *bring your own criteria* functionality that enables users to provide the desired policy to be verified. As shown in Figure 9, we convert the policy yaml format to the *bring your own criteria format*. In this way, it is easy to enforce the policy by identifying violations using Granite Guardian 4.1.

7 Conclusions

Policy specification is a key factor in ensuring compliance and governance in GenAI applications. There is not a standardized way to specify these policies in a way that is suitable for human experts to provide guidance and in a way that at the same time can influence model alignment and verification. To close this gap, we presented Granite.trust Policy Tools for specifying, testing, and enforcing safety policies in GenAI applications. Our contributions address different stages of the GenAI lifecycle:

1. The **policy schema** enables human-readable yet machine-enforceable policy specification.
2. The **synthetic data pipeline** translates policies into training data for model alignment.
3. The **exception-based policy tracking** enables governance in single and multi-agent systems by tracking policy violations.
4. The **complementary tools** facilitate policy definition, conflict detection, and compliance verification.

Together, these enable organizations to define their safety requirements once and enforce them throughout their GenAI stack. We have made our tools and example policies available as open source:

https://github.com/ibm-granite/granite.trust.policy-tools

We welcome new ideas, contributions and feedback.

8 Acknowledgments

We want to thank Pavan Kapanipathi, Kshitij Fadnis, and Siva Sankalp for all their help integrating our pipeline into the DGT framework. We also want to thank Kristi Spess, John McBroom, Bryan Bortnick, Derek Leist, Betsy Greytok, Shafiq Abedin and Ismael Faro for their feedback.

References

[1] Swapnaja Achintalwar, Ioana Baldini, Djallel Bouneffouf, Joan Byamugisha, Maria Chang, Pierre Dognin, Eitan Farchi, Ndivhuwo Makondo, Aleksandra Mojsilović, Manish Nagireddy, et al. Alignment studio: Aligning large language models to particular contextual regulations. *IEEE Internet Computing*, 28(5):28–36, 2024.

[2] Christopher Alberts, Audrey Dorofee, James Stevens, and Carol Woody. Introduction to the octave approach. *Carnegie Mellon Software Engineering Institute*, 2003.

[3] Anthropic. Claude's new constitution. https://www.anthropic.com/news/claude-new-constitution, 2026.

[4] Zahra Ashktorab, Alessandra Buccella, Jason D'Cruz, Zoë Fowler, Andrew Gill, Kei Yan Leung, PD Magnus, John Richards, and Kush R Varshney. Who's sorry now: User preferences among rote, empathic, and explanatory apologies from llm chatbots. *ACM Transactions on Computer-Human Interaction*, 2025.

[5] Frank Bagehorn, Kristina Brimijoin, Elizabeth M Daly, Jessica He, Michael Hind, Luis Garces-Erice, Christopher Giblin, Ioana Giurgiu, Jacquelyn Martino, Rahul Nair, et al. Ai risk atlas: taxonomy and tooling for navigating ai risks and resources. *arXiv preprint arXiv:2503.05780*, 2025.

[6] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[7] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[8] Faeze Brahman, Sachin Kumar, Vidhisha Balachandran, Pradeep Dasigi, Valentina Pyatkin, Abhilasha Ravichander, Sarah Wiegreffe, Nouha Dziri, Khyathi Chandu, Jack Hessel, et al. The art of saying no: Contextual noncompliance in language models. *Advances in Neural Information Processing Systems*, 37:49706–49748, 2024.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[10] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

[11] Fei Ding and Baiqiao Wang. Improved supervised fine-tuning for large language models to mitigate catastrophic forgetting. *arXiv preprint arXiv:2506.09428*, 2025.

[12] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3029–3051, 2023.

[13] Ian W Eisenberg, Lucía Gamboa, and Eli Sherman. The unified control framework: Establishing a common foundation for enterprise ai governance, risk management and regulatory compliance. *arXiv preprint arXiv:2503.05937*, 2025.

[14] Cloud Native Computing Foundation. Oscal compass. https://github.com/oscal-compass, 2026.

[15] Kai Fronsdal, Isha Gupta, Abhay Sheshadri, Jonathan Michala, Stephen McAleer, Rowan Wang, Sara Price, and Sam Bowman. Petri: Parallel exploration of risky interactions. https://github.com/safety-research/petri, 2025.

[16] Iker García-Ferrero, David Montero, and Roman Orus. Refusal steering: Fine-grained control over llm refusal behaviour for sensitive topics. *arXiv preprint arXiv:2512.16602*, 2025.

[17] Shaona Ghosh, Prasoon Varshney, Makesh Narsimhan Sreedhar, Aishwarya Padmakumar, Traian Rebedea, Jibin Rajan Varghese, and Christopher Parisien. AEGIS2.0: A diverse AI safety dataset and risks taxonomy for alignment of LLM guardrails. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5992–6026, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.

[18] Google. Gemma 4: Our most intelligent open models, built from gemini 3 research and technology to maximize intelligence-per-parameter. https://deepmind.google/models/gemma/gemma-4/, 2026.

[19] Google. Model alignment of gemma. https://colab.research.google.com/github/pair-code/model-alignment/blob/main/notebooks/Gemma_for_Model_Alignment.ipynb, 2026.

[20] Google. Shieldgemma 2. https://ai.google.dev/gemma/docs/shieldgemma, 2026.

[21] Kristjan Greenewald, Luis Lastras, Thomas Parnell, Vraj Shah, Lucian Popa, Giulio Zizzo, Chulaka Gunasekara, Ambrish Rawat, and David Cox. Activated lora: Fine-tuned llms for intrinsics. *arXiv preprint arXiv:2504.12397*, 2025.

[22] Lukas Helff, Felix Friedrich, Manuel Brack, Patrick Schramowski, and Kristian Kersting. Llavaguard: Vlm-based safeguard for vision dataset curation and safety assessment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8322–8326, 2024.

[23] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[24] IBM. Ares. https://github.com/IBM/ares, 2026.

[25] IBM. Fms-dgt: Dgt (pronounced "digit") is a framework that enables different algorithms and models to be used to generate synthetic data. https://github.com/IBM/fms-dgt, 2026.

[26] IBM. Granite guardian 4.1. https://huggingface.co/ibm-granite/granite-guardian-4.1-8b#example-4-requirement-checking-judging-instruction-following, 2026.

[27] IBM. Granite models. https://huggingface.co/ibm-granite, 2026.

[28] ISO. Iso/iec 42001:2023 information technology — artificial intelligence — management system. https://www.iso.org/standard/42001, 2026.

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

[30] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[31] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in neural information processing systems*, 36:51991–52008, 2023.

[32] Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, et al. Synthetic data (almost) from scratch: Generalized instruction tuning for language models. *arXiv preprint arXiv:2402.13064*, 2024.

[33] Sijia Liu, Yang Liu, and Nathalie Baracaldo. *Machine Unlearning for Governance of Foundation Models*. Springer, 2026.

[34] Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu, Yuguang Yao, Hang Li, Kush R Varshney, et al. Rethinking machine unlearning for large language models. *arXiv preprint arXiv:2402.08787*, 2024.

[35] Meta. How-to guides: Evaluations. https://www.llama.com/docs/how-to-guides/evaluations/, 2026.

[36] Meta. Purple llama. https://github.com/meta-llama/PurpleLlama, 2026.

[37] MIT. Mit ai risk repository. https://airisk.mit.edu/, 2026.

[38] MLCommons. Mlcommons: Ailuminate safety benchmark. https://mlcommons.org/ailuminate/safety/, 2025. Accessed: 2025-11-12.

[39] Gary D. Lopez Munoz, Amanda J. Minnich, Roman Lutz, Richard Lundeen, Raja Sekhar Rao Dheekonda, Nina Chikanov, Bolor-Erdene Jagdagdorj, Martin Pouliot, Shiven Chawla, Whitney Maxwell, Blake Bullwinkel, Katherine Pratt, Joris de Gruyter, Charlotte Siska, Pete Bryan, Tori Westerhoff, Chang Kawaguchi, Christian Seifert, Ram Shankar Siva Kumar, and Yonatan Zunger. Pyrit: A framework for security risk identification and red teaming in generative ai systems. https://arxiv.org/abs/2410.02828, 2024.

[40] OpenAI. Openai model spec. https://model-spec.openai.com/2025-12-18.html, 2025.

[41] OWASP. Owasp top 10 for large language model applications. https://genai.owasp.org/llm-top-10/, 2025.

[42] Inkit Padhi, Manish Nagireddy, Giandomenico Cornacchia, Subhajit Chaudhury, Tejaswini Pedapati, Pierre Dognin, Keerthiram Murugesan, Erik Miehling, Martín Santillán Cooper, Kieran Fraser, Giulio Zizzo, Muhammad Zaid Hameed, Mark Purcell, Michael Desmond, Qian Pan, Zahra Ashktorab, Inge Vejsbjerg, Elizabeth M. Daly, Michael Hind, Werner Geyer, Ambrish Rawat, Kush R. Varshney, and Prasanna Sattigeri. Granite guardian. https://arxiv.org/abs/2412.07724, 2024.

[43] Inkit Padhi, Karthikeyan Natesan Ramamurthy, Prasanna Sattigeri, Manish Nagireddy, Pierre Dognin, and Kush R Varshney. Value alignment from unstructured text. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1083–1095, 2024.

[44] Swetasudha Panda, Naveen Jafer Nizar, and Michael L Wick. Llm improvement for jailbreak defense: Analysis through the lens of over-refusal. In *Neurips Safe Generative AI Workshop 2024*, 2024.

[45] Alicia Parrish, Angelica Chen, Nikita Nangia, Vishakh Padmakumar, Jason Phang, Jana Thompson, Phu Mon Htut, and Samuel Bowman. Bbq: A hand-built bias benchmark for question answering. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2086–2105, 2022.

[46] Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! *arXiv preprint arXiv:2310.03693*, 2023.

[47] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[48] Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. Xstest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400, 2024.

[49] Mark Russinovich, Ahmed Salem, and Ronen Eldan. Great, now write an article about that: The crescendo {Multi-Turn}{LLM} jailbreak attack. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2421–2440, 2025.

[50] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. *https://arxiv.org/abs/2402.03300*, 2(3):5, 2024.

[51] Peter Slattery, Alexander K Saeri, Emily AC Grundy, Jess Graham, Michael Noetel, Risto Uuk, James Dao, Soroush Pour, Stephen Casper, and Neil Thompson. The ai risk repository: A comprehensive meta-review, database, and taxonomy of risks from artificial intelligence. *arXiv preprint arXiv:2408.12622*, 2024.

[52] Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36:2511–2565, 2023.

[53] HELM Safety team. Helm safety leaderboard. https://crfm.stanford.edu/helm/safety/latest/, 2025. Accessed: 2025-11-14.

[54] Changsheng Wang, Chongyu Fan, Yihua Zhang, Jinghan Jia, Dennis Wei, Parikshit Ram, Nathalie Baracaldo, and Sijia Liu. Rethinking unlearning for large reasoning models. In *ICML 2025 Workshop on Machine Unlearning for Generative AI*, 2025.

[55] Changsheng Wang, Yihua Zhang, Jinghan Jia, Parikshit Ram, Dennis Wei, Yuguang Yao, Soumyadeep Pal, Nathalie Baracaldo, and Sijia Liu. Invariance makes llm unlearning resilient even to unanticipated downstream fine-tuning. *arXiv preprint arXiv:2506.01339*, 2025.

[56] Rowan Wang, Avery Griffin, Johannes Treutlein, Ethan Perez, Julian Michael, Fabien Roger, and Sam Marks. Modifying llm beliefs with synthetic document finetuning. https://alignment.anthropic.com/2025/modifying-beliefs-via-sdf/, 2025.

[57] Xinran Wang, Qi Le, Ammar Ahmed, Enmao Diao, Yi Zhou, Nathalie Baracaldo, Jie Ding, and Ali Anwar. Map: Multi-human-value alignment palette. *arXiv preprint arXiv:2410.19198*, 2024.

[58] Zifeng Wang, Chun-Liang Li, Vincent Perot, Long Le, Jin Miao, Zizhao Zhang, Chen-Yu Lee, and Tomas Pfister. Codeclm: Aligning language models with tailored synthetic data. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3712–3729, 2024.

[59] Cameron Wolfe. Understanding and using supervised fine-tuning (sft) for language models. https://cameronrwolfe.substack.com/p/understanding-and-using-supervised, 2023.

[60] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*, 2024.

[61] Da Yin, Xiao Liu, Fan Yin, Ming Zhong, Hritik Bansal, Jiawei Han, and Kai-Wei Chang. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4031–4047, 2023.

[62] Yi Zeng, Kevin Klyman, Andy Zhou, Yu Yang, Minzhou Pan, Ruoxi Jia, Dawn Song, Percy Liang, and Bo Li. Ai risk categorization decoded (air 2024): From government regulations to corporate policies. *arXiv preprint arXiv:2406.17864*, 2024.

[63] Yi Zeng, Yu Yang, Andy Zhou, Jeffrey Ziwei Tan, Yuheng Tu, Yifan Mai, Kevin Klyman, Minzhou Pan, Ruoxi Jia, Dawn Song, et al. Air-bench 2024: A safety benchmark based on risk categories from regulations and policies. *arXiv preprint arXiv:2407.17436*, 2024.