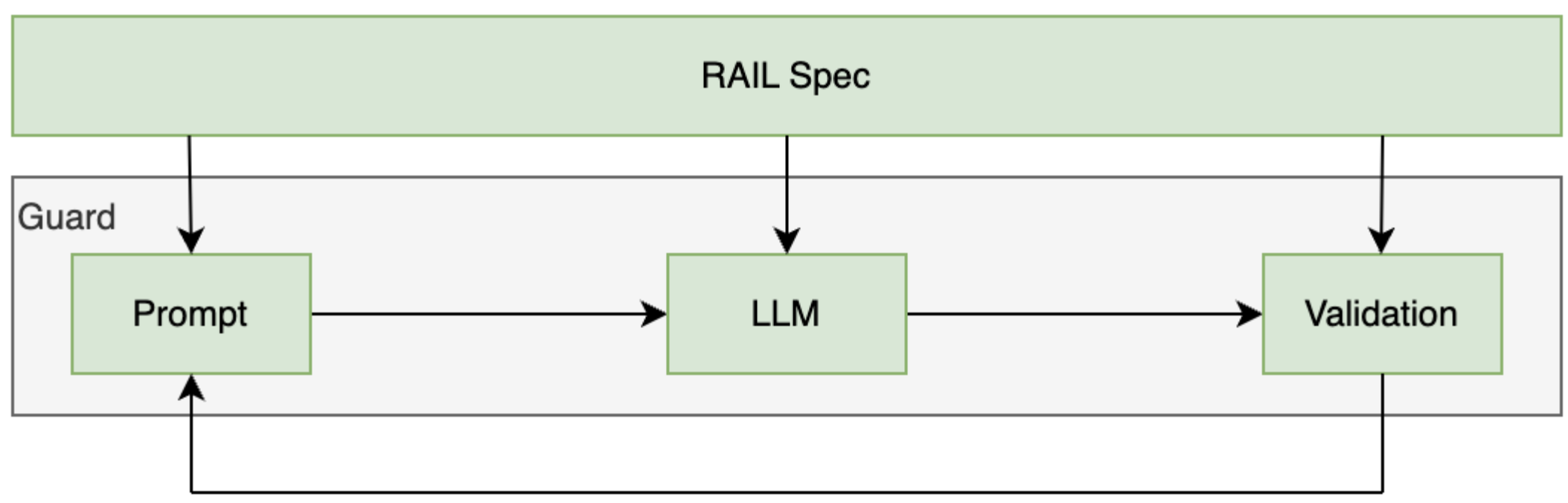
What is GuardrailsAI

A leading open-source framework to define and enforce assurance for LLM applications

- Specification Language
- Seamless Orchestration
- Comprehensive Validators Library
- Custom Validations Framework

How GuardrailsAI works



- RAIL Spec (**R**eliable **AI** Markup **L**anguage)
  - Comprises of output definition and prompt template
  - Compiles into a prompt for interaction with LLM
- Guard - Enforces Rail Spec standards consistently

- Guardrails
  - Rail Spec + Guard Object
  - Applies validations on LLM output and makes corrections
  - Ensures quality and ethical standards are maintained

GuardrailsAI – RAIL Spec

The RAIL spec defines prompts, output structures, validation standards, and AI corrections, using XML, Pydantic, or strings for guards. It comprises of output schema and prompt to be passed to LLM.

```
<rail version="0.1">
    <output>
        <object name="billing_info">
            <string name="patient_id" description="Unique identifier for the patient" />
            <object name="insurance_details">
                <string name="provider" description="Name of the insurance provider" />
                <string name="policy_number" description="Insurance policy number" />
            </object>
            <list
                name="billing_items"
                description="List of services billed, including the service name, cost, and whether it's covered by insurance.">
                <object>
                    <string name="service" description="Name of the medical service provided" />
                    <float name="cost" description="Cost of the service" format="valid-range: 0-10000" />
                </object>
            </list>
            <float name="total_cost" description="Total cost of all services rendered" format="valid-range: 0 100000" />
            <float name="insurance_coverage" description="Total amount covered by insurance" format="valid-range: 0 100000" />
            <float name="out_of_pocket" description="Total amount the patient needs to pay out-of-pocket" format="valid-range: 0
            100000" />
        </object>
    </output>
    <prompt>
        Given the details of a patient's billing inquiry, please compile a JSON object that accurately reflects the requested billing
        information.
        ${billing_inquiry_notes}
        ${gr.complete_json_suffix_v2}
    </prompt>
</rail>
```

GuardrailsAI – Output

Defines AI model's expected output structure, ensuring data integrity and quality through precise criteria.

- Structure
  - Allows detailed output schema customization
  - Enables key-value pair structured data generation
  - Facilitates complex nested structure outputs
  - Supports lists for flexible value collections
- Data Integrity
  - Enforces specific format and quality standards
  - Implements corrective actions for quality assurance

```
<output>
    <object name="billing_info">
        <string name="patient_id" description="Unique identifier for the patient" />
        <list
            name="billing_items"
            description="List of services billed, including the service name, cost, and whether it's covered by insurance.">
            <object>
                <string name="service" description="Name of the medical service provided" />
                <float name="cost" description="Cost of the service" format="valid-range: 0-10000" />
            </object>
        </list>
        <float
            name="total_cost"
            description="Total cost of all services rendered"
            format="min-val: 0"
            on-fail-min-val="exception"
        />
        <float name="insurance_coverage" description="Total amount covered by insurance" format="valid-range: 0 100000" />
        <float name="out_of_pocket" description="Total amount the patient needs to pay out-of-pocket" format="valid-range: 0 100000"
        on-fail-min-val="exception"/>
    </object>
</output>
```

GuardrailsAI – Validators

Validators are automated checkpoints that assess AI-generated outputs against specific criteria, triggering predefined corrective actions upon failure.

- Automated output validation against criteria
- Executes corrective actions on failures
- Supports custom validation functions/classes
- Utilizes runtime parameters for dynamic checks

```
<float
    name="total_cost"
    description="Total cost of all services rendered"
    format="min-val: 0"
    on-fail-min-val="exception"
/>
```

GuardrailsAI – Built-inValidators

- ValidRange: Validates value within specified range (e.g., age: 18-65).
- SimilarToList: Ensures similarity to known values (e.g., colors: “red,” “blue”).
- EndpointIsReachable: Verifies reachable URL (e.g., API endpoint).
- ProvenanceV0: Compares LLM text to source (e.g., paraphrased sentence).
- ProvenanceV1: Validates LLM output against source.
- BugFreePython: Ensures bug-free Python code.
- ExtractiveSummary: Validates summaries.
- EndsWith: Checks value ending (e.g., filename: “.txt”).
- BugFreeSQL: Verifies correct SQL queries.
- RegexMatch: Validates pattern match (e.g., email format).
- PIIFilter**:** Filters sensitive info (e.g., SSN).
- QARelevanceLLMEval**:** Assesses answer relevance. (e.g.:answers matching context)
- UpperCase**:** Ensures all uppercase. (e.g.: “hello” to “HELLO”)
- IsHighQualityTranslation**:** Validates translation quality. (e.g. Accuracy of English-to-French translation)
- ReadingTime**:** Estimates reading time. (e.g. reading time for an article)
- ValidURL**:** Checks valid URL. (e.g. “https://example.com” is valid)
- ToxicLanguage: Detects toxic / offensive content.
- SqlColumnPresence: Verifies SQL columns. e.g.: Presence of columns.
- OneLine: Fits within a single line.
- CompetitorCheck: Compares with competitors.
- OnTopic: Validates on-topic content.
- ValidChoices: Checks valid options. (e.g. Check with predefined choices.)
- SaliencyCheck: Assesses content importance. E.g. (key points in a summary)
- ExcludeSqlPredicates: Prevents SQL issue. (e.g. Avoiding unsafe SQL queries)
- TwoWords: Requires at least two words. (e.g. “apple pie.”)
- DetectSecrets: Filters sensitive data. (e.g. Removing passwords)
- ValidLength: Length within limits. (e.g. Checking if a tweet fits character limit)
- PydanticFieldValidator: Validates fields. (e.g. Ensuring valid input)
- LowerCase: All lowercase. (e.g. “HELLO” to “hello.”)
- SimilarToDocument: Similarity to reference. (e.g. Comparing generated summary to an article.)
- ExtractedSummarySentencesMatch: Summary accuracy. (e.g. Confirming extracted sentences match input.)
- IsProfanityFree: Filters profanity. (e.g.: Detecting inappropriate language.)
- RemoveRedundantSentences**:** Removes redundant sentences. (e.g. Eliminating repeated information in a paragraph.)

Reference – Guardrails AI Validators

GuardrailsAI – Validator OnFail Policies

Action	Behavior
reask	Reask the LLM to generate an output that meets the quality criteria. The prompt used for reasking contains information about which quality criteria failed, which is auto-generated by the validator.
fix	Programmatically fix the generated output to meet the quality criteria. E.g. for the formatter two-words, the programatic fix simply takes the first 2 words of the generated string.
filter	Filter the incorrect value. This only filters the field that fails, and will return the rest of the generated output.
refrain	Refrain from returning an output. If a formatter has the corrective action refrain, then on failure there will be a None output returned instead of the JSON.
noop	Do nothing. The failure will still be recorded in the logs, but no corrective action will be taken.
exception	Raise an exception when validation fails.
fix_reask	First, fix the generated output deterministically, and then rerun validation with the deterministically fixed output. If validation fails, then perform reasking.

GuardrailsAI – Prompt

A set of instructions provided to LLMs guiding content generation and analysis with dynamic parameters and structured formatting for compliant outputs.

```
<prompt>
    Given the details of a patient's billing inquiry, please compile a
    JSON object that accurately reflects the requested billing
    information.                  <!-- (1)! -->
    ${billing_inquiry_notes}      <!-- (2)! -->
    ${gr.complete_json_suffix_v2} <!-- (3)! -->
</prompt>
```

1. High-level instructions to guide LLM
2. Runtime parameters substituted for customization
3. Formatting guidelines using Boilerplate text to ensure valid JSON formatting

```
<prompt>
    Given the details of a patient's billing inquiry, please compile a JSON object that accurately reflects
    the requested billing information.  <!-- (1)! -->

    A 56-year-old male patient, who has been managing a persistent facial and scalp rash characterized by
    macules—most severe in the beard, eyebrows, and nostrils area—seeks billing information for his recent
    treatments. The patient reports moderate improvement after using an over-the-counter steroid cream. He
    inquires about the charges for the consultation, any diagnostic tests conducted, and the cost of
    recommended treatments, including the steroid cream. Additionally, he seeks clarification on what
    portion of these costs are covered by his insurance plan. <!-- (2)! -->

    Refer to the XML schema provided below to understand the structure of the information that needs to be
    extracted from the billing inquiry.  <!-- (3)! -->

    ${output_schema}

    ONLY return a valid JSON object (no other text is necessary), where the key of the field in JSON is the
    `name` attribute of the corresponding XML, and the value is of the type specified by the corresponding
    XML's tag. The JSON MUST conform to the XML format, including any types and format requests e.g.
    requests for lists, objects and specific types. Be correct and concise. If you are unsure anywhere,
    enter `null`.

    Here are examples of simple (XML, JSON) pairs that show the expected behavior:
    - `<![CDATA[<string name='foo' format='two-words lower-case' />`]]> => `{'foo': 'example one'}`
    - `<![CDATA[<list name='bar'><string format='upper-case' /></list>]]>` => `{"bar": ['STRING ONE',
    'STRING TWO', etc.]}`
    - `<![CDATA[<object name='baz'><string name="foo" format="capitalize two-words" /><integer name="index"
    format="1-indexed" /></object>]]>` => `{'baz': {'foo': 'Some String', 'index': 1}}`
</prompt>
```

GuardrailsAI – Guard

Primary interface for orchestrating interactions with AI engines, utilizing RAIL specifications to validate the LLM output and perform necessary actions.

Two main flows

1. call

```python
from guardrails import Guard
import openai

guard = Guard.from_rail(...)

# Call the LLM wrapped in Guardrails
raw_output, validated_output, *rest = guard(
    openai.chat.completions.create,
    "prompt_params",
    max_tokens=1024,
    temperature=0.3,
)

print(raw_output)
print(validated_output)
```

2. validate

```python
from guardrails import Guard
import openai

guard = Guard.from_rail(...)

# Using Guard to validate LLM output
validated_output = Guard.validate(llm_output="validate this llm output")

print(validated_output)
```

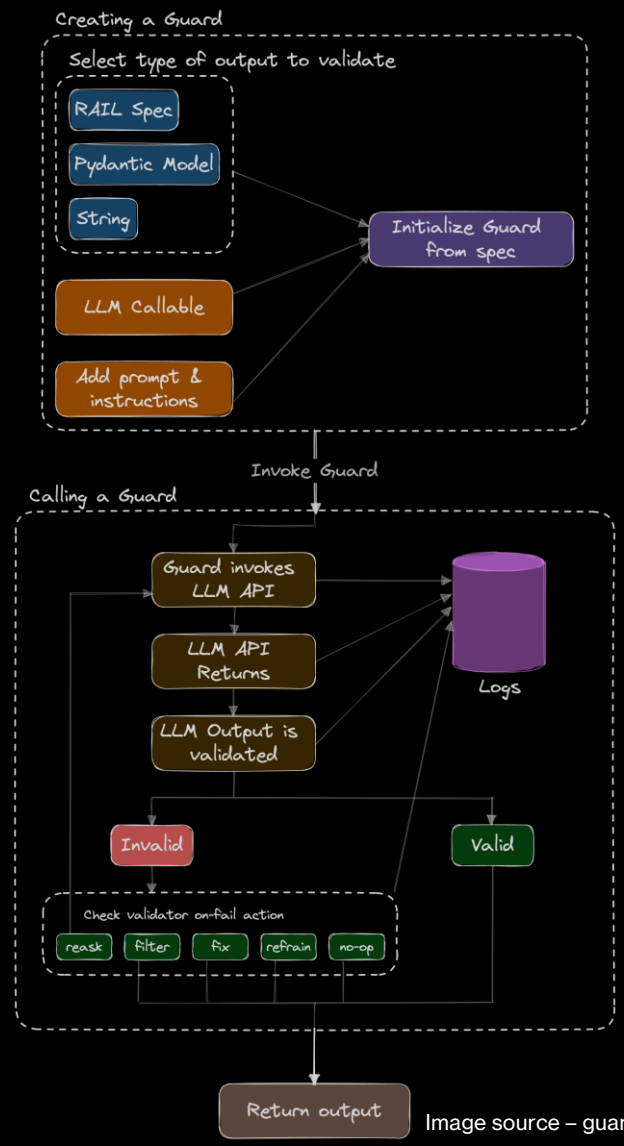


Image source – guardrailsai.com

Error Handling and Retries

- Automatic retries for common errors (e.g. connection timeout, ratelimit, serviceunavailable)
- Robust error handling for common LLM issues
- May not catch all possible errors and developers must handle exceptions not covered in the implementation

What GuardrailsAI cannot help with

- LLMs cannot gain new abilities via Guardrails AI
- Unable to fix LLMs' inherent bias or discrimination
- Not a universal fix for LLM issues