
A Tester's Role in
Building AI Systems

- **Quality Engineering Architect & AI Engineer**
- Former Head of Engineering & AI
- **ML/AI for 7+ years!**
- Test Automation University

Adobe

QA at the Point

Name that movie!

Name that movie!

Predict how much customers would spend

Number of people	Purchase Amount
1	**10**
2	**20**
4	**40**

How much would 3 people spend?

Testers can
Analyze, Assess, and Evaluate
Problems and Requirements

Prevent overstocking and understocking to maximize profits and reduce waste

Number of people	Purchase Amount
1	**10**
2	**20**
4	**40**

What data do we need now?


ROCK ENGLISH
FISH & CHIPS
1 CASEMATES SQUARE
GIBRALTAR
TEL +350 20051218
REG 23-05-2016 13:33
CLERK 4 MC#01 237313
CT 1
TABLE No: 107
1 LRG COD & CHIPS £11.95
1 REG COD & CHIPS £7.55
1 BREAD & BUTTER £1.00
3 SOFT DRINK £4.50
TOTAL... £25.00
(€37.50)

Receipt Analyzer

Transform your receipts into actionable insights with AI-powered analysis

Smart Receipt Analysis

Upload your receipt and let our AI extract every detail

Click to change image

receipt-2.png
2.18 MB

Analyze Receipt

Analysis Results

Total Amount
GBP25

Merchant
Rock English Fish & Chips

Date
2016-05-23

Receipt Details

Receipt #	237313
Time	13:33

Expense Category

Primary Category	Food & Dining
Sub-category	Quick Service / Fast Food
Business Deductible	Yes

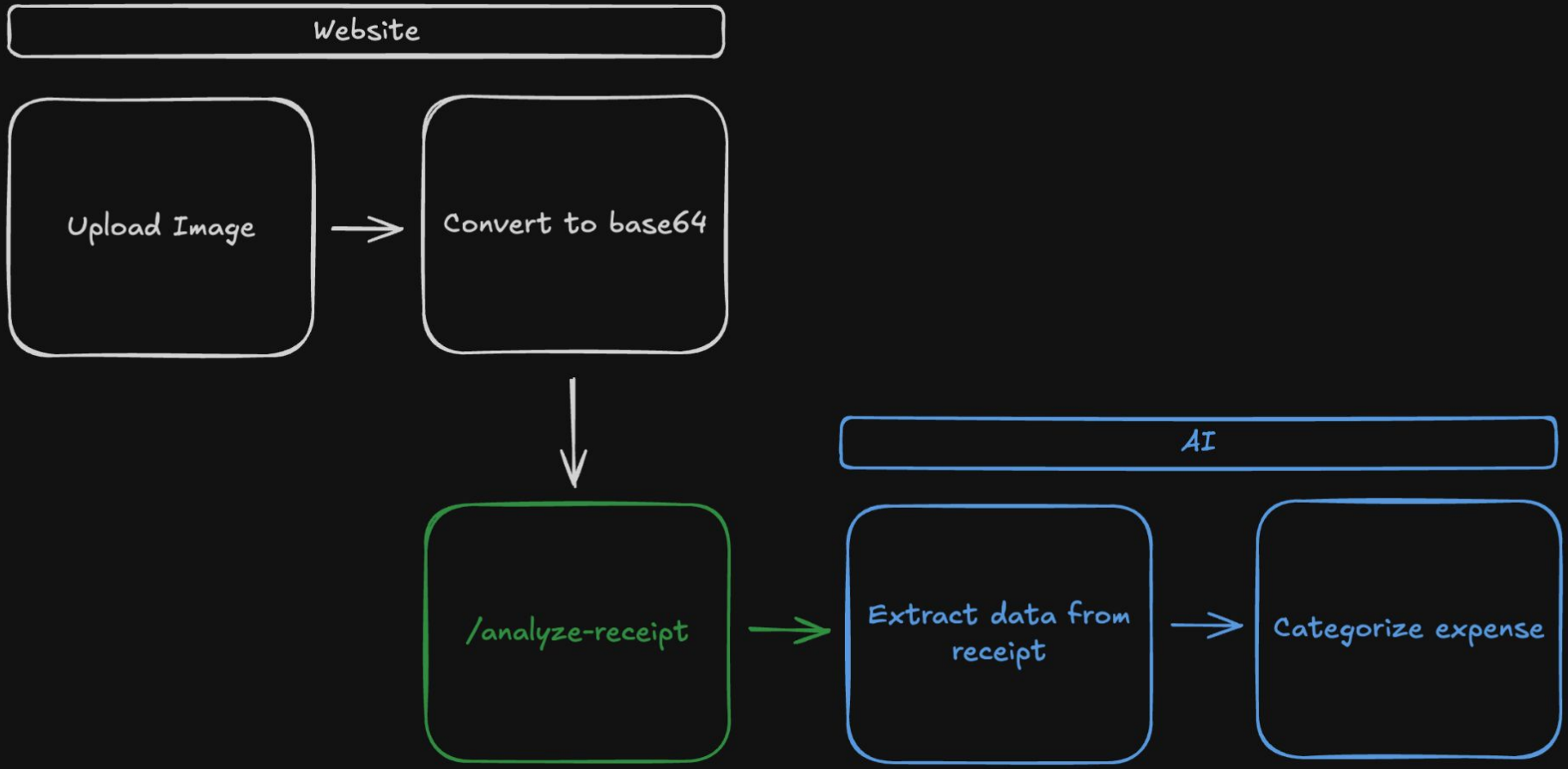
Tags

food · takeout · fast food · restaurant · fish and chips

Itemized Breakdown

Item	Qty	Price	Total
Lrg Cod & Chips	1	$11.95	**$11.95**
Reg Cod & Chips	1	$7.55	**$7.55**
Bread & Butter	1	$1.00	**$1.00**
Soft Drink	3	$1.50	**$4.50**

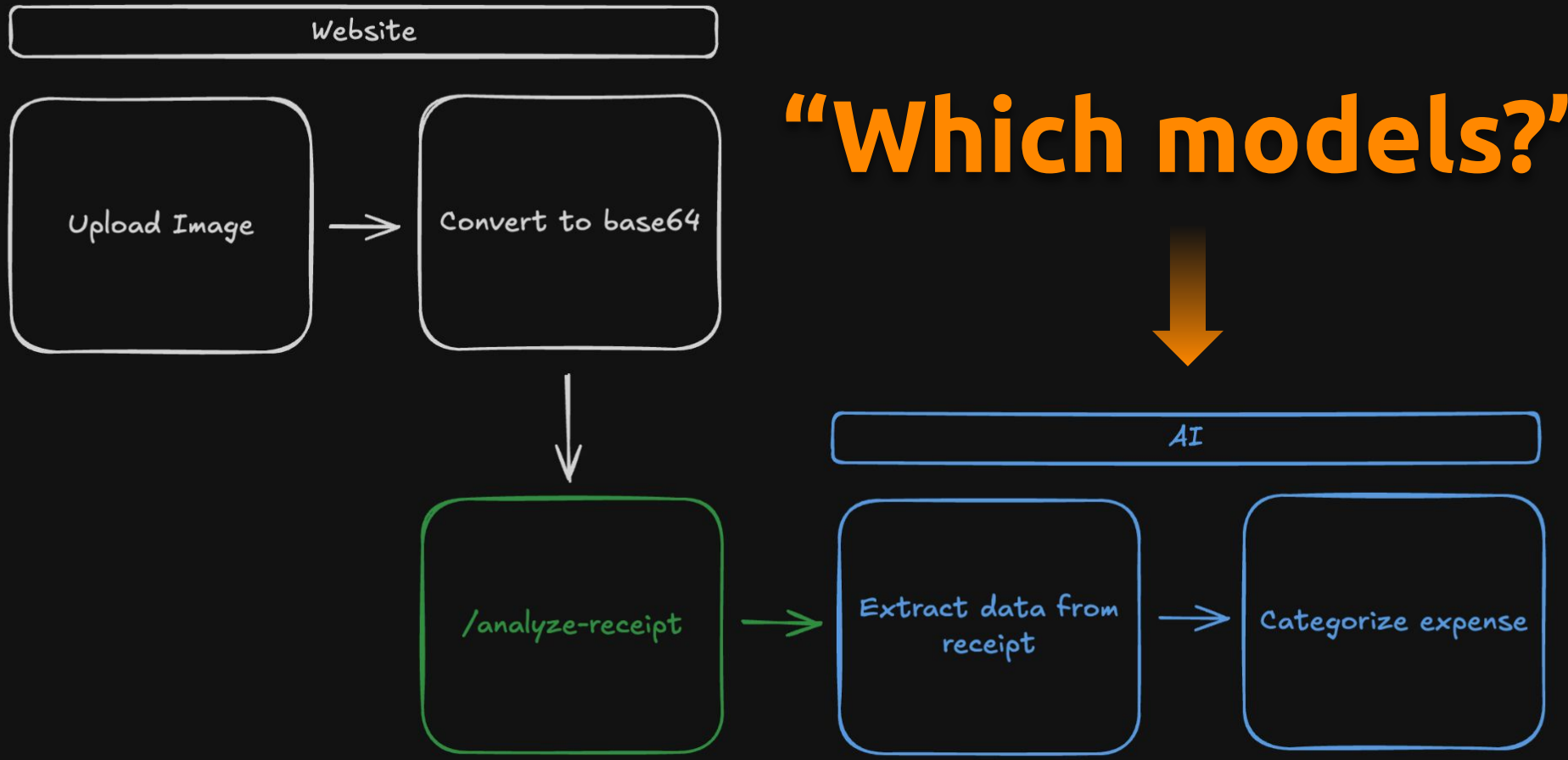
Analysis completed on 9/18/2025, 10:26:42 PM • Confidence: high

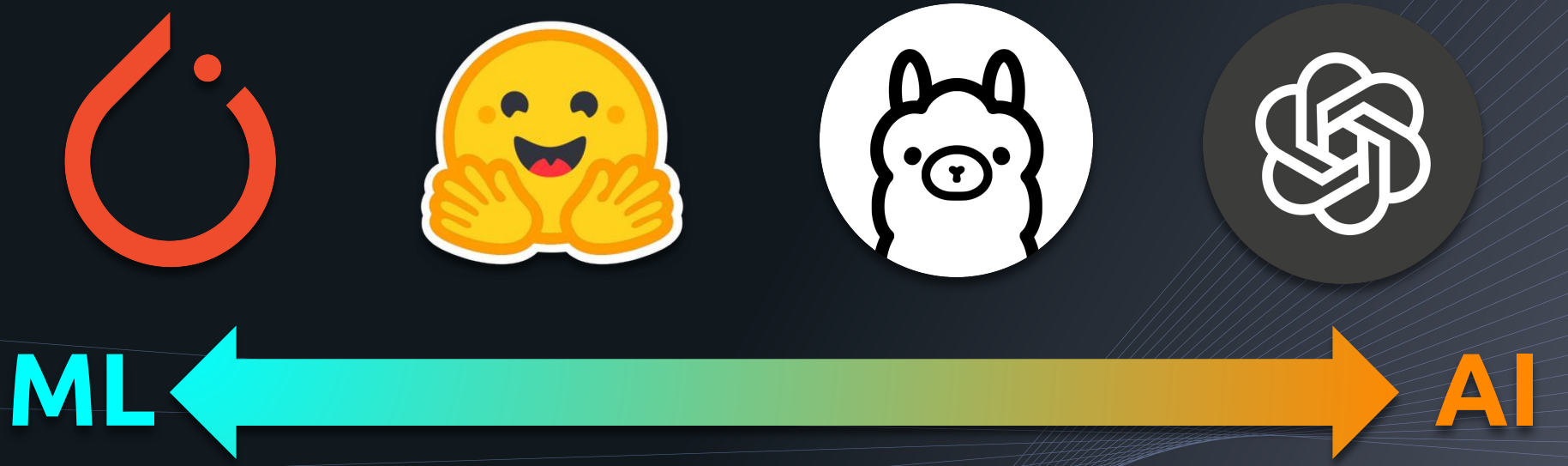

Website
Upload Image
Convert to base64
/analyze-receipt
AI
Extract data from receipt
Categorize expense

Name that movie!

Name that movie!

Testers can add
questions, risks and scenarios
to Technical Designs


Website
Upload Image
Convert to base64
/analyze-receipt
"Which models?"
AI
Extract data from receipt
Categorize expense


ML
AI

ibm-granite/granite-docling-258M

Image-Text-to-Text | Transformers | Safetensors | 4 datasets | English | idefics3 | image-to-text | text-generation | documents | code | formula | chart | ocr | layout | table | document-parse | docling | granite | extraction | math | conversational | arxiv:2501.17887 | arxiv:2503.11576 | arxiv:2305.03393 | License: apache-2.0

Edit model card

granite-docling-258m

Granite Docling is a multimodal Image-Text-to-Text model engineered for efficient document conversion. It preserves the core features of Docling while maintaining seamless integration with DoclingDocuments to ensure full compatibility.

Downloads last month
16,405

Safetensors

Model size 258M params | Tensor type BF16

Chat template | Files info

Inference Providers NEW

Image-Text-to-Text

This model isn't deployed | 10 Ask for provider support

Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion

Nikolaos Livathinos *, Christoph Auer *, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panagiotis Vagenas, Cesar Berrospi, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, Peter W. J. Staar

IBM Research, Rüschlikon, Switzerland

Please send correspondence to: deepsearch-core@zurich.ibm.com

Abstract

We introduce *Docling*, an easy-to-use, self-contained, MIT-licensed, open-source toolkit for document conversion, that can parse several types of popular document formats into a unified, richly structured representation. It is powered by state-of-the-art specialized AI models for layout analysis (DocLayNet) and table structure recognition (TableFormer), and runs efficiently on commodity hardware in a small resource budget. Docling is released as a Python package and can be used as a Python API or as a CLI tool. Docling's modular architecture and efficient document representation make it easy to implement extensions, new features, models, and customizations. Docling has been already integrated in other popular open-source frameworks (e.g., LangChain, LlamaIndex, spaCy), making it a natural fit for the processing of documents and the development of high-end applications. The open-source community has fully engaged in using, promoting, and developing for Docling, which gathered 10k stars on GitHub in less than a month and was reported as the No. 1 trending repository in GitHub worldwide in November 2024.

Repository — https://github.com/DS4SD/docling

1 Introduction

Converting documents back into a unified machine-processable format has been a major challenge for decades due to their huge variability in formats, weak standardization and printing-optimized characteristic, which often discards structural features and metadata. With the advent of LLMs and popular application patterns such as retrieval-augmented generation (RAG), leveraging the rich content embedded in PDFs, Office documents, and scanned document images has become ever more relevant. In the past decade, several powerful document understanding solutions have emerged on the market, most of which are commercial software, SaaS offerings on hyperscalers (Auer et al. 2022) and most recently, multimodal vision-language models. Typically, they incur a cost (e.g., for licensing or LLM inference) and cannot be run easily on local hardware. Meanwhile, only a handful of different open-source tools cover PDF, MS Word, MS PowerPoint, Images, or HTML conversion, leaving a significant feature and quality gap to proprietary solutions.

With *Docling*, we recently open-sourced a very capable and efficient document conversion tool which builds on the powerful, specialized AI models and datasets for layout analysis and table structure recognition that we developed and presented in the recent past (Livathinos et al. 2021; Pfitzmann et al. 2022; Lysak et al. 2023). Docling is designed as a simple, self-contained Python library with permissive MIT license, running entirely locally on commodity hardware. Its code architecture allows for easy extensibility and addition of new features and models. Since its launch in July 2024, Docling has attracted considerable attention in the AI developer community and ranks top on GitHub's monthly trending repositories with more than 10,000 stars at the time of writing. On October 16, 2024, Docling reached a major milestone with version 2, introducing several new features and concepts, which we outline in this updated technical report, along with details on its architecture, conversion speed benchmarks, and comparisons to other open-source assets.

The following list summarizes the features currently available on Docling:

- Parses common document formats (PDF, Images, MS Office formats, HTML) and exports to Markdown, JSON, and HTML.
- Applies advanced AI for document understanding, including detailed page layout, OCR, reading order, figure extraction, and table structure recognition.
- Establishes a unified `DoclingDocument` data model for rich document representation and operations.
- Provides fully local execution capabilities making it suitable for sensitive data and air-gapped environments.
- Has an ecosystem of plug-and-play integrations with prominent generative AI development frameworks, including LangChain and LlamaIndex.
- Can leverage hardware accelerators such as GPUs.

2 State of the Art

Document conversion is a well-established field with numerous solutions already available on the market. These solutions can be categorized along several key dimensions, including open vs. closed source, permissive vs. restrictive licensing, Web APIs vs. local code deployment, susceptibility

*These authors contributed equally.

Docling: An Efficient Open-Source Toolkit for AI-driven Document Conversion

Nikolaos Livathinos *, Christoph Auer *, Matsym Lysak, Ahmed Nassar, Michele Dolfi, Panagiotis Vagenas, Cesar Berrospi, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valery Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, Peter W. J. Staar

IBM Research, Rüschlikon, Switzerland

Please send correspondence to: deepsearch-core@zurich.ibm.com

Abstract

We introduce Docling , an easy-to-use, self-contained, MITlicensed, open-source toolkit for document conversion, that can parse several types of popular document formats into a unified, richly structured representation. It is powered by state-of-the-art specialized AI models for layout analysis (DocLayNet) and table structure recognition (TableFormer), and runs efficiently on commodity hardware in a small resource budget. Docling is released as a Python package and can be used as a Python API or as a CLI tool. Docling's modular architecture and efficient document representation make it easy to implement extensions, new features, models, and customizations. Docling has been already integrated in other popular open-source frameworks (e.g., LangChain, LlamaIndex, spaCy), making it a natural fit for the processing of documents and the development of high-end applications. The open-source community has fully engaged in using, promoting, and developing for Docling, which gathered 10k stars on GitHub in less than a month and was reported as the No. 1 trending repository in GitHub worldwide in November 2024.

Repository - https://github.com/DS4SD/docling

1 Introduction

Converting documents back into a unified machinerecessable format has been a major challenge for decades due to their huge variability in formats, weak standardization and printing-optimized characteristic, which often discards structural features and metadata. With the advent of LLMs and popular application patterns such as retrieval-augmented generation (RAG), leveraging the rich content embedded in PDFs, Office documents, and scanned document images has become ever more relevant. In the past decade, several powerful document understanding solutions have emerged on the market, most of which are commercial software, SaaS offerings on hyperscalers (Auer et al. 2022) and most recently, multimodal vision-language models. Typically, they incur a cost (e.g., for licensing or LLM inference) and cannot be run easily on local hardware. Meanwhile, only a handful of different open-source tools cover PDF, MS Word, MS PowerPoint, Images, or HTML conversion, leaving a significant feature and quality gap to proprietary solutions.

*These authors contributed equally.

With Docling , we recently open-sourced a very capable and efficient document conversion tool which builds on the powerful, specialized AI models and datasets for layout analysis and table structure recognition that we developed and presented in the recent past (Livathinos et al. 2021; Pfitzmann et al. 2022; Lysak et al. 2023). Docling is designed as a simple, self-contained Python library with permissive MIT license, running entirely locally on commodity hardware. Its code architecture allows for easy extensibility and addition of new features and models. Since its launch in July 2024, Docling has attracted considerable attention in the AI developer community and ranks top on GitHub's monthly trending repositories with more than 10,000 stars at the time of writing. On October 16, 2024, Docling reached a major milestone with version 2, introducing several new features and concepts, which we outline in this updated technical report, along with details on its architecture, conversion speed benchmarks, and comparisons to other open-source assets.

The following list summarizes the features currently available on Docling:

- · Parses common document formats (PDF, Images, MS Office formats, HTML) and exports to Markdown, JSON, and HTML.
- · Applies advanced AI for document understanding, including detailed page layout, OCR, reading order, figure extraction, and table structure recognition.
- · Establishes a unified DoclingDocument data model for rich document representation and operations.
- · Provides fully local execution capabilities making it suitable for sensitive data and air-gapped environments.
- · Has an ecosystem of plug-and-play integrations with prominent generative AI development frameworks, including LangChain and Llamandex.


PDF
PPTX
DOCX
HTML
docling
Docling
{DOC}
JSON,
MD,
Figures
...
Chunking,
LlamaIndex,
LangChain
Your
GenAI App

Testers can understand data models and contracts


```python
from datetime import datetime
from typing import Decimal, List, Optional
from pydantic import BaseModel, Field


class ReceiptItem(BaseModel):
    """Model for individual receipt items"""

    name: str = Field(description="Name of the item")
    quantity: int = Field(description="Quantity purchased")
    price: Decimal = Field(description="Individual item price")
    total: Decimal = Field(description="Total cost for this item")
```

POST /analyze-receipt Analyze Receipt

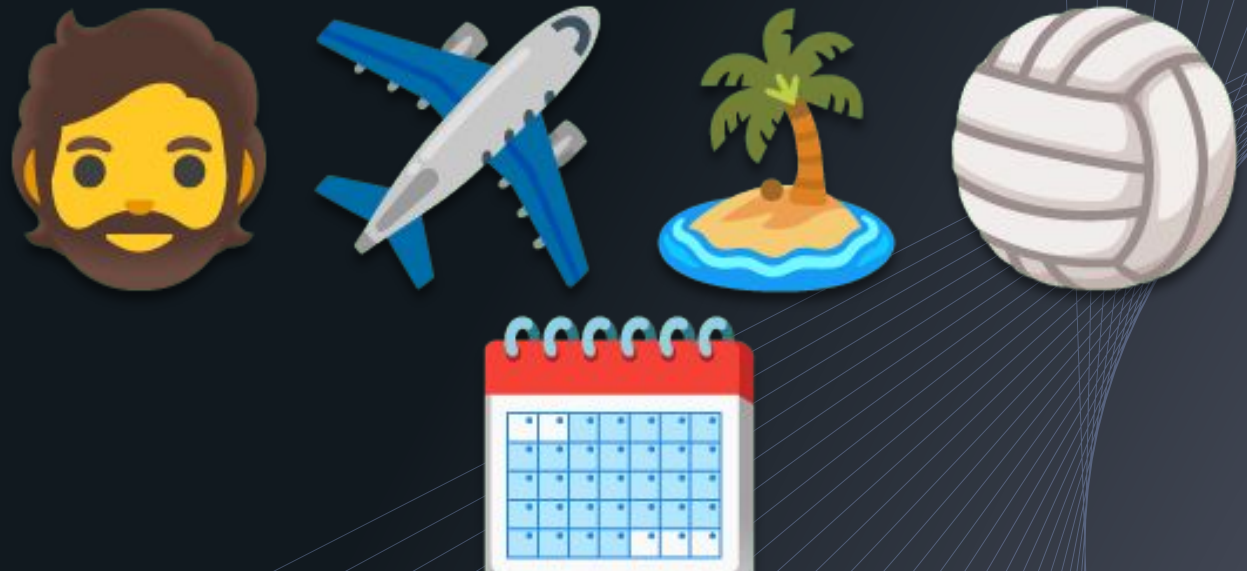
Responses

Code	Description	Links
200	Successful Response Media type application/json Controls Accept header. **Example Value** | Schema	*No links*


```
{
  "merchant_name": "string",
  "date": "string",
  "time": "string",
  "total_amount": "string",
  "currency": "string",
  "tax_amount": "string",
  "items": [
    {
      "name": "string",
      "quantity": 0,
      "price": "string",
      "total": "string"
    }
  ],
```

Testers can
leverage their existing skills

Name that movie!



Name that movie!

Name that movie!

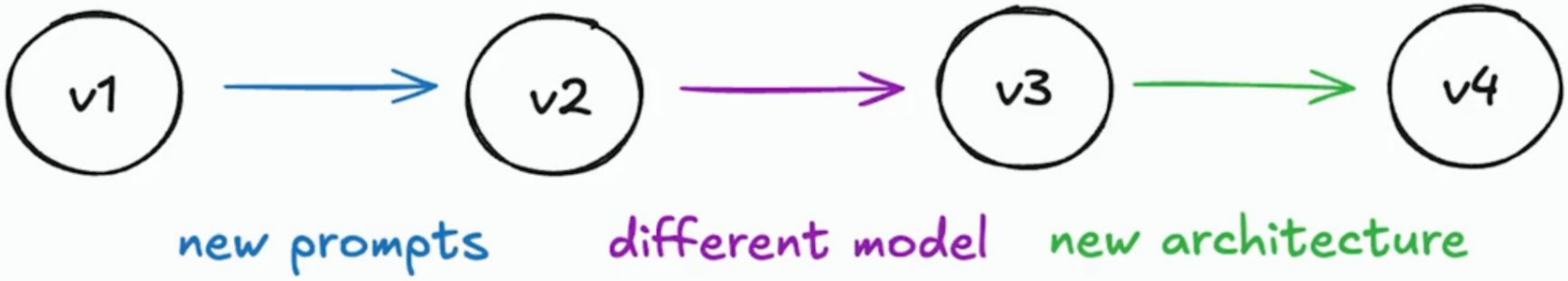
Name that movie!

google/BIG-bench/emoji_movie


```
{
  "input": "What movie does this emoji describe? 👨‍🚀🕐🌎👩‍🔬",
  "target_scores": {
    "interstellar": 1,
    "alice's adventures in wonderland": 0,
    "the sound of music": 0,
    "planet of the apes": 0,
    "toy story": 0
  },
  "target": "interstellar"
},
```

```python
EMOJI_EXAMPLES = [
    {
        "input": "👨‍🚀🕰️🌎👩‍🔬",
        "output": "interstellar"
    },
    {
        "input": "🤡🎈",
        "output": "it"
    },
]

@pytest.mark.parametrize("example", EMOJI_EXAMPLES)
def test_emoji(example):
    response = graph.ainvoke(example["input"])
    assert response["output"] == example["output"]
```



Offline Evaluation: To test and evaluate our application, we need **datasets**

Dataset A

Example 1	Input=x1	Output=y1
Example 2	Input=x2	Output=y2
Example 3	Input=x3	Output=y3
Example 4	Input=x4	Output=y4

...

Looks like a Test Suite!


```
{
  "merchant_name": "Rock English Fish & Chips",
  "date": "2016-05-23",
  "time": "13:33",
  "total_amount": 25.00,
  "currency": "GBP",
  "items": [
    {
      "name": "LRG COD & CHIPS",
      "quantity": 1,
      "price": 11.95,
      "total": 11.95
    },
    {
      "name": "REG COD & CHIPS",
      "quantity": 1,
      "price": 7.55,
      "total": 7.55
    },
    {
      "name": "BREAD & BUTTER",
      "quantity": 1,
      "price": 1,
      "total": 1
    }
```

Testers can
strategize testing and evals
(we *really* need help here!)

Types of Feedback

Categorical

- is_correct (yes, no)
- type_of_input (integer, number, string)
- is_helpful (yes, no)
- did_succeed (succeeded, failed)

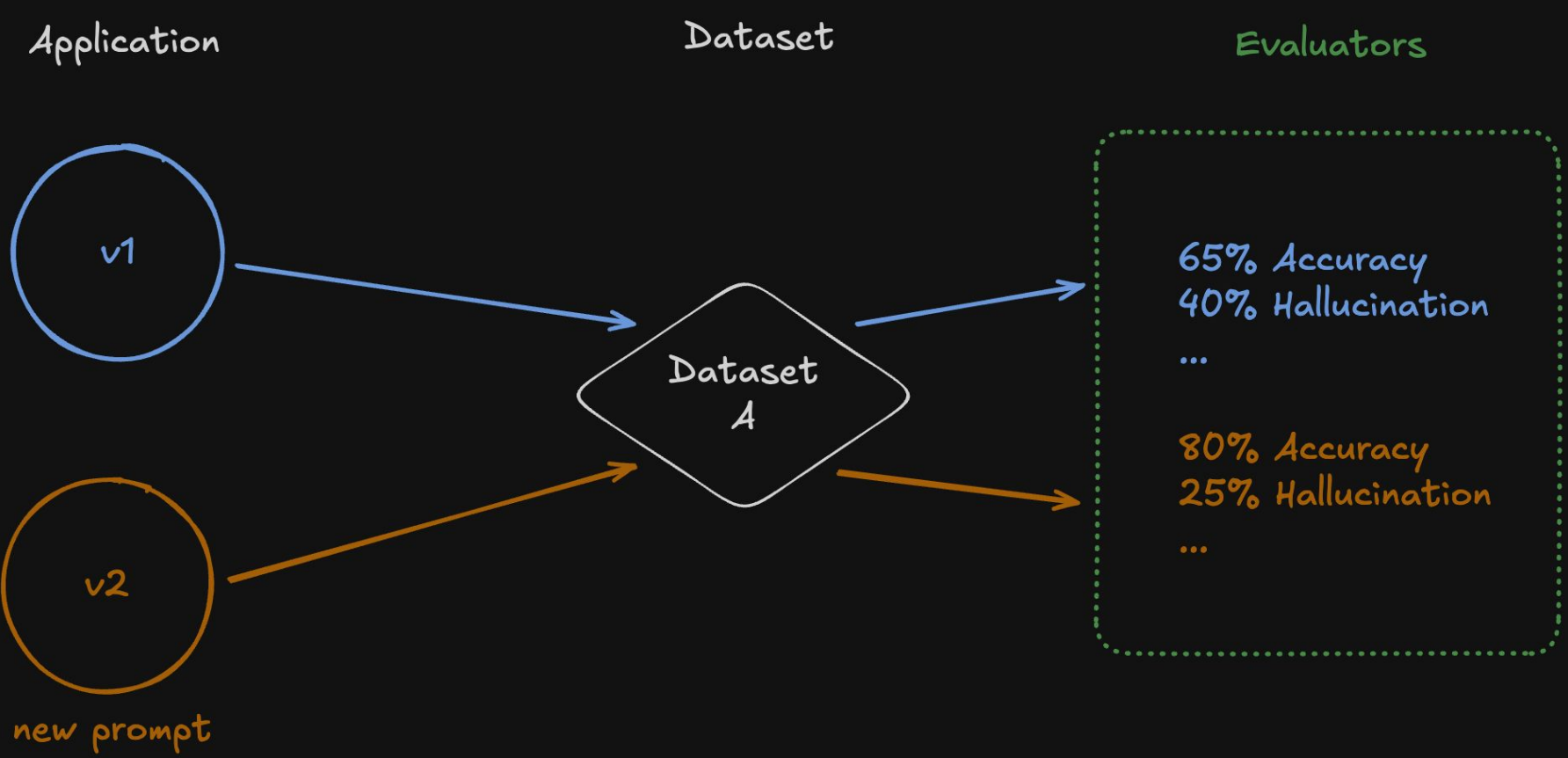
Continuous

- correctness 7 (1 - 10)
- similarity 3 (1 - 5)
- helpfulness 3 (1 - 10)

Attribute

↳ **Metric** → **Measurement**

↳ **...**


Application
Dataset
Evaluators
v1
v2
new prompt
Dataset
A
65% Accuracy
40% Hallucination
...
80% Accuracy
25% Hallucination
...

Types of Evaluators

- Annotations = Manual
- Custom Code = Traditional Code Logic
- LLM-as-Judge = AI Evaluation
- Pairwise = Head-to-head preference
- Composite = summary of many evaluators

```
  {
    "name": "Large Cod & Chips",
    "quantity": 1,
    "price": "11.95",
    "total": "11.95"
  },
  {
    "name": "Regular Cod & Chips",
    "quantity": 1,
    "price": "7.55",
    "total": "7.55"
  },
  {
    "name": "Bread & Butter",
    "quantity": 1,
```


JSON RAW

Receipt Number

Address

Phone

Category

restaurant

Annotation Rubric

Instructions

Review the receipt (merchant name, line items, etc) and complete each Feedback Item.

Feedback

+ Feedback

category_is_relevant is_relevant

Determine whether the AI's given `category` is relevant to the items on the receipt or not.

- is_not_relevant — The `category` is NOT relevant
- is_relevant — The `category` is relevant

extraction_accuracy 10.00

On a scale of 0 - 10, where 0 means the AI got everything wrong and 10 means the AI got everything right, score how accurate the AI extraction was.

Min: 0, Max: 10

10

Reviewer Notes

Add your note... N

Add Custom Code Evaluator

Add a new Custom Code Evaluator for the current dataset

Save

Evaluator Name

is_accurate

Apply to past runs

Function

Write the code in Python

All standard Python libraries are supported, along with a few external libraries like numpy and jsonschema. See docs for full list.

```python
def perform_eval(run, example):
    want = example["outputs"]
    got = run["outputs"]

    if got == want:
        return {
          "score": 1,
          "feedback": "Values match"
        }
    else:
        return {
          "score": 0,
          "feedback": "Values don't match"
        }
```

Input Run


```
{
  "outputs": {
    "date": "2016-05-23",
    "time": "13:33",
    "items": [
      {
        "name": "Large Cod & Chips",
        "price": 11.95,
        "total": 11.95,
        "quantity": 1
```


Input Example


```
{
  "outputs": {
    "date": "2016-05-23",
    "time": "13:33",
    "items": [
      {
        "name": "Large Cod & Chips",
        "price": 11.95,
        "total": 11.95,
        "quantity": 1
```


Result


```
{
  "score": 1,
  "feedback": "Values match"
}
```

Create Evaluator

Discard changes | Create

HUMAN

Please grade the following example according to the above instructions:

<example>
<input>
input ×
</input>

<output>
output ×
</output>

<reference_outputs>
referenceOutput ×
</reference_outputs>
</example>

Feedback configuration

Define your evaluation criteria. Describe what you're measuring, then select a response format. This configuration structures how your evaluation results are returned.

hallucination

Include reasoning | Advanced

Description

TRUE if the output contains any hallucinations (unsupported claims, contradictions, speculative details, or inaccurate facts). FALSE if all claims are directly verifiable from the input context.

Response Format Boolean

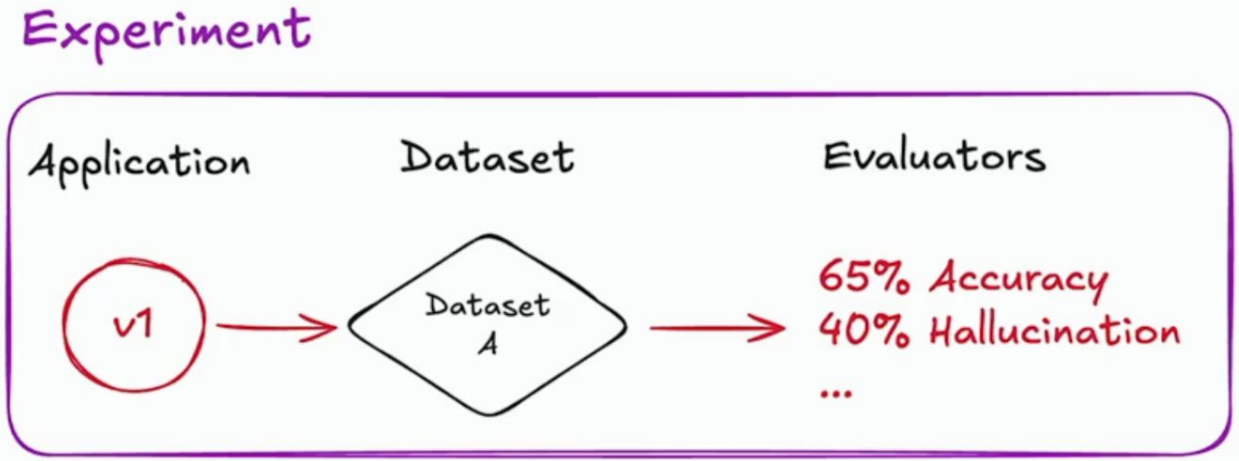
REFERENCE OUTPUT


```
{
  "date": "2016-05-23",
  "time": "13:33",
  "items": [
    {
      "name": "Large Cod & Chips",
      "price": 11.95,
      "total": 11.95,
      "quantity": 1
    }
```

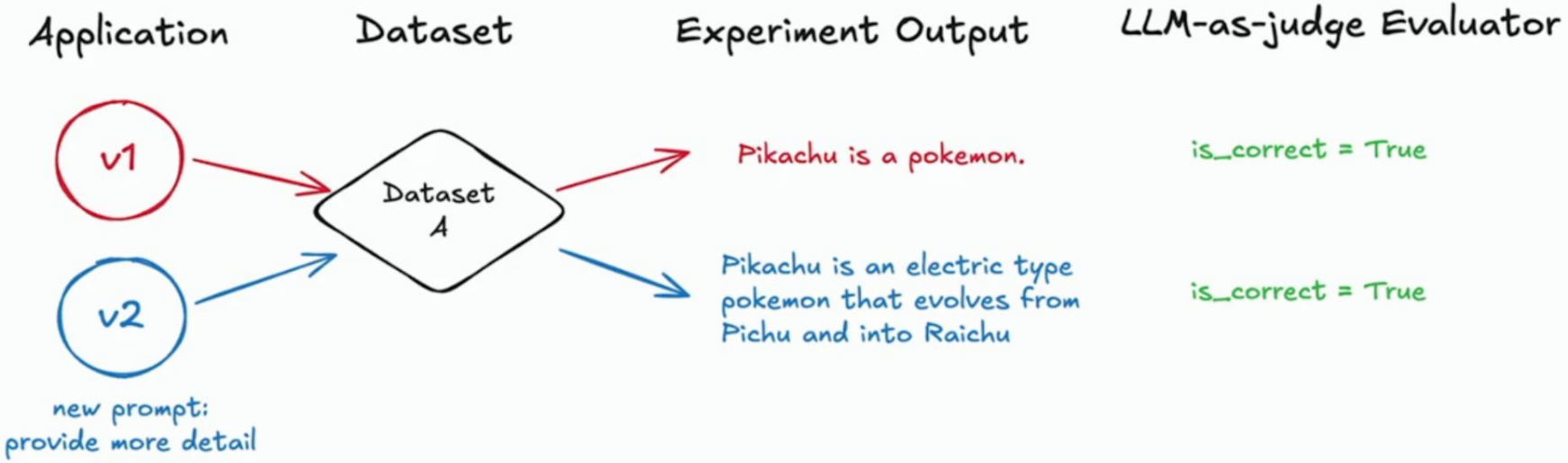

OUTPUT

Run an experiment via the SDK or in the Playground to generate outputs. This is recommended before creating an evaluator to aid with mapping variables from your prompt to the output.

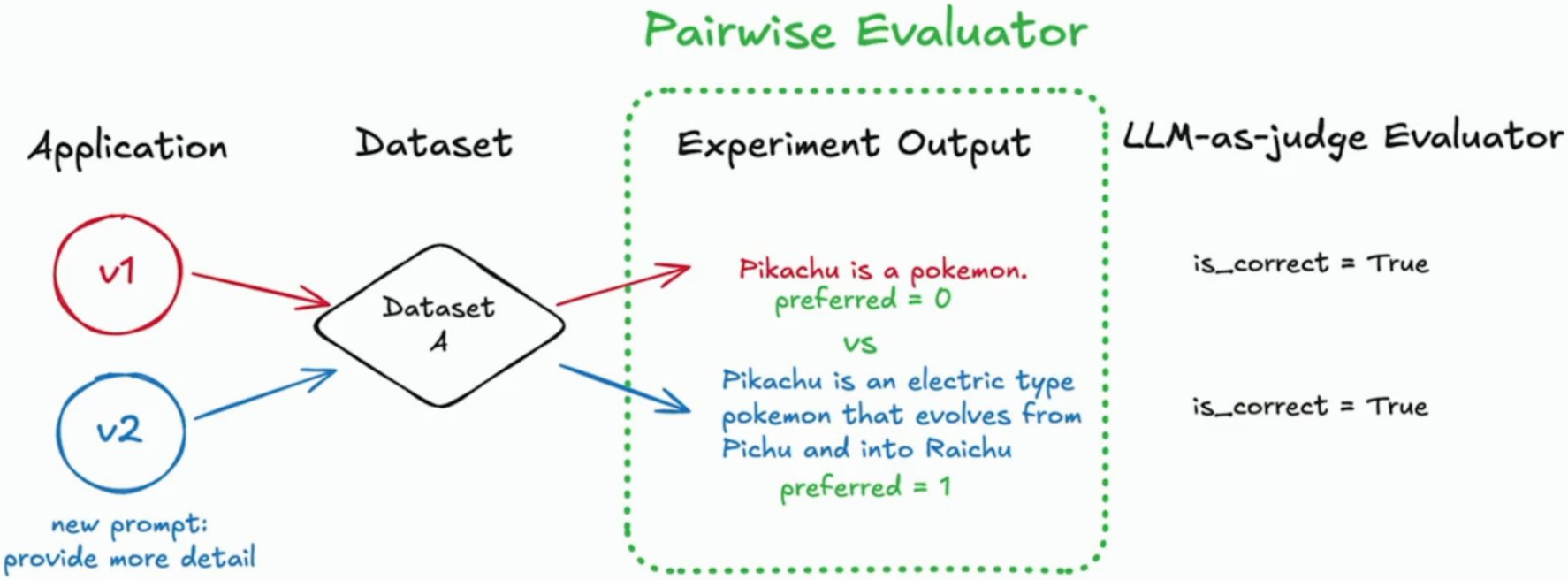
Experiment: Running your **application** over a **dataset**, and **evaluating** performance



Pairwise Evaluation



Pairwise Evaluation



Viewing Pairwise Experiment

Diff Display

Good Summarizer-bafea4ec vs. Bad Summarizer-06ff299d-e509 compared by ranked_preference where a higher score indicates better performance

	Good Summarizer-bafea4ec 5 0	Bad Summarizer-06ff299d 0 5
	{ "output": "In the meeting, Mr. Patel expressed interest in purchasing a midsize sedan for his daily commute and was particularly interested in a hybrid model due to his 50-mile daily drive. Bob recommended the Ford Fusion Hybrid, which fit Mr. Patel's budget of $30,000, and after a smooth test drive, they discussed financing options. The deal was closed with an agreement on 0% financing for 36 months, making Mr. Patel a new Ford owner." } ranked_preference 1.00 summary_score 5.00 2.19s SUCCESS 0	{ "output": "Bob successfully closed a deal with Mr. Patel on a Ford Fusion Hybrid after offering a test drive and 0% financing for 36 months, meeting his budget and driving preferences." } ranked_preference 0.00 summary_score 5.00 1.37s SUCCESS 0
	{ "output": "Bob and Ms. Thompson met at Ford Motors where Ms. Thompson expressed interest in browsing for an SUV with a potential purchase next year. Bob introduced her to the Ford Escape and offered to email her the latest offers for future consideration. Although no deal was made, Ms. Thompson was open to further communication, indicating potential for a future purchase." } ranked_preference 1.00 summary_score 5.00 1.64s SUCCESS 0	{ "output": "Bob and Ms. Thompson had a friendly conversation at Ford Motors, where she expressed interest in possibly buying an SUV next year, leading Bob to offer to send her information and keep in touch for future inquiries." } ranked_preference 0.00 summary_score 5.00 3.86s SUCCESS 0

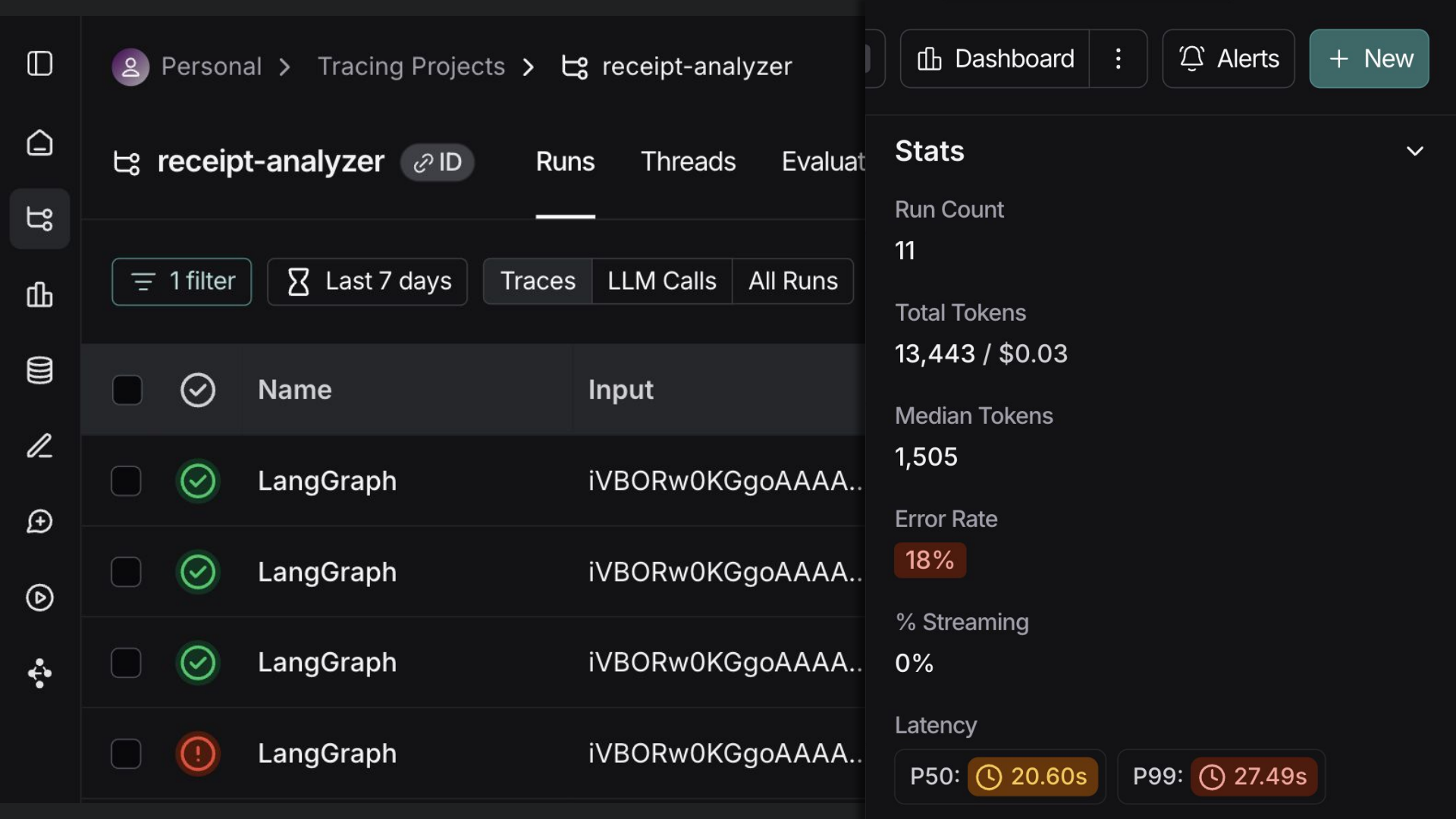
Types of Evaluators

- Annotations = Manual
- Custom Code = Traditional Code Logic
- LLM-as-Judge = AI Evaluation
- Pairwise = Head-to-head preference

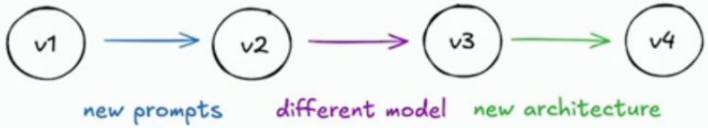
You can define evaluators in your code or in the LangSmith UI

Testers can
observe and measure quality
(we need help here, too)

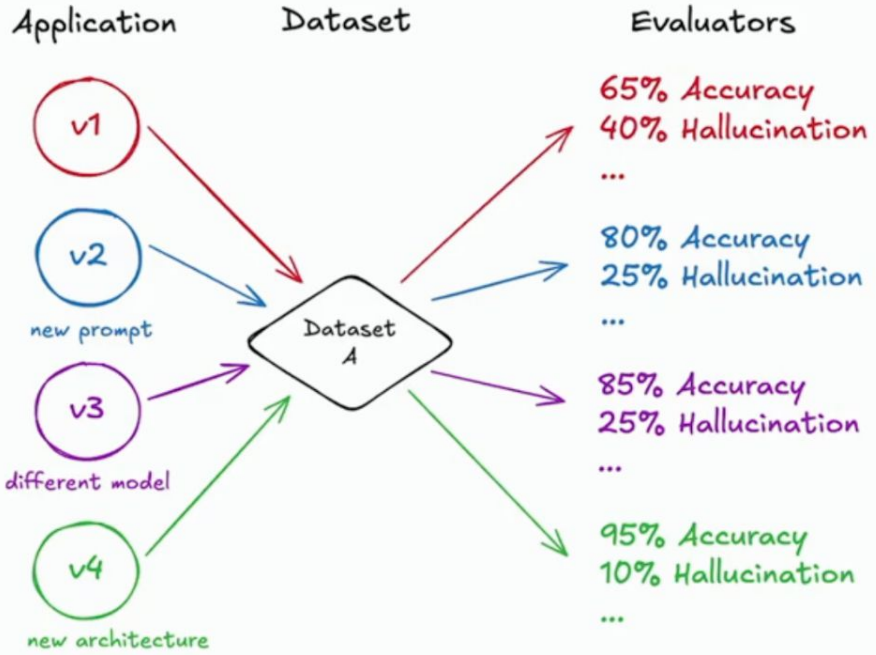
Input	Output	Error	Start Time	Latency	Dataset	Annotation Queue	Tokens	Cost
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/22/2025, 9:58:46...	18.54s			2,933	$0.0049291
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/22/2025, 9:49:23...	27.51s			3,323	$0.0080179
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/22/2025, 12:45:10...	24.49s			3,460	$0.0072248
iVBORw0KGgoAAAA...		TypeError("mode...	9/22/2025, 12:38:17...	27.34s			3,727	$0.0097955
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/18/2025, 10:26:22...	24.69s			0	
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/18/2025, 2:21:57 ...	21.59s			0	
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/17/2025, 11:52:14 ...	17.47s			0	
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/17/2025, 11:42:40...	18.55s			0	
iVBORw0KGgoAAAA...		AttributeError("'f...	9/17/2025, 11:16:20 ...	0.08s			0	
iVBORw0KGgoAAAA...	iVBORw0KGgoAA...		9/17/2025, 11:10:09 ...	20.60s			0	


Personal
Tracing Projects
receipt-analyzer
Dashboard
Alerts
New
receipt-analyzer
ID
Runs
Threads
Evaluat
1 filter
Last 7 days
Traces
LLM Calls
All Runs
Name
Input
LangGraph
iVBORw0KGgoAAAA..
LangGraph
iVBORw0KGgoAAAA..
LangGraph
iVBORw0KGgoAAAA..
LangGraph
iVBORw0KGgoAAAA..
Stats
Run Count
11
Total Tokens
13,443 / $0.03
Median Tokens
1,505
Error Rate
18%
% Streaming
0%
Latency
P50: 20.60s
P99: 27.49s

Without Testing



With Testing and Evaluation



Model Attributes

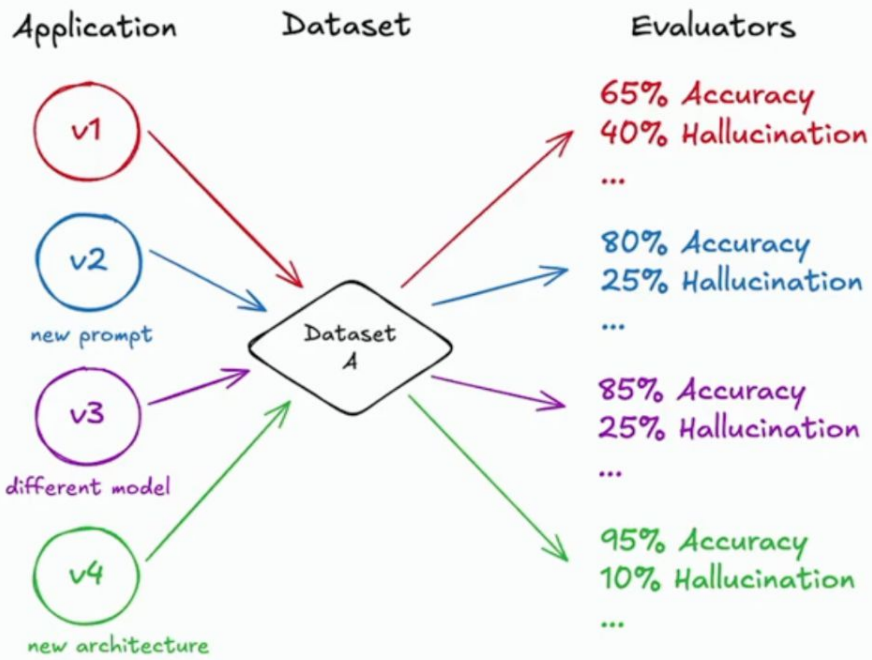
	GPT 3.5	GPT 4o	GPT 4o-mini	Mistral 7B	Llama3.1 70B	Llama3.2 3B
Latency	1.9s	5.6s	3.5s	2.2s	5.1s	1.3s
Tokens	1161	1175	1165	361	1014	994
Cost	0.0002$	0.0018$	0.0008$	0.0$	0.0$	0.0$

Quality Attributes	GPT 3.5	GPT 4o	GPT 4o-mini	Mistral 7B	Llama3.1 70B	Llama3.2 3B
Hallucinations	7%	3%	4%	63%	6%	18%
Accuracy	95%	100%	99%	73%	96%	87%
Reliability	90%	99%	93%	50%	87%	75%

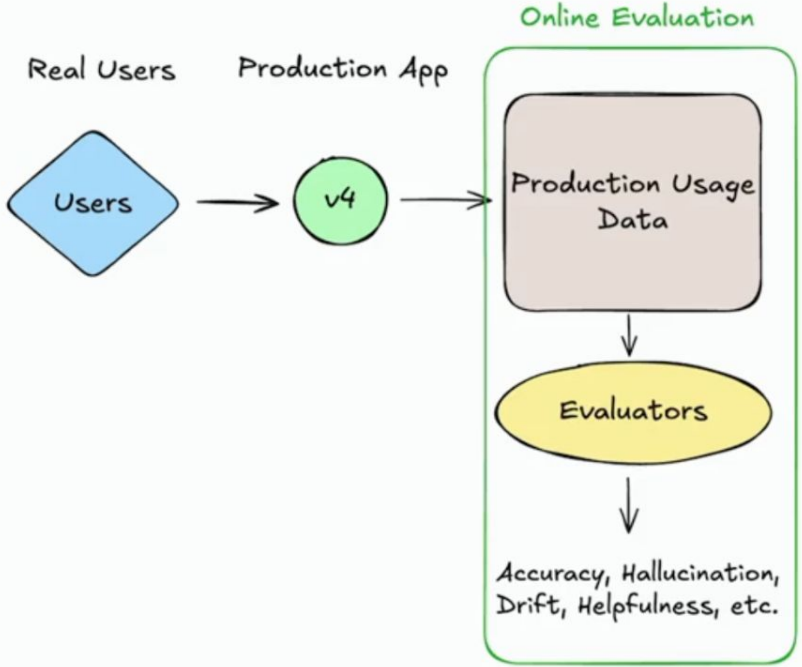
Offline and Online Evaluation

Online Evaluation involves evaluating real production performance with end users

Offline Evaluation



Online Evaluation



Testers can
rise to the challenge!

Get started:
academy.langchain.com/courses/intro-to-langsmith

LET'S CONNECT!

Come find me and ask questions!

@CarlosKidman