Evaluating model performance Beta

Copy page

Test and improve model outputs through evaluations.

Evaluations (often called **evals**) help you develop high-quality LLM applications by ensuring that the results you generate with a model meet accuracy criteria that you specify. Similar to having unit tests set up for traditional software, having good evals in place for your application lets you try new models and iterate on your prompts with greater confidence.

The Evaulations tooling in the OpenAI dashboard can help you create and run evals for your application. In this guide, you'll learn how evals work in general and how you can use the Evaluations tool in the dashboard to test your prompts and LLM outputs.

0:00 / 5:17

Understand your objective

Before you begin, you should understand the behavior you want from the model. Here are some questions you should be able to answer:

- What kind of output do I want the model to generate?
- What kinds of inputs should the model be able to handle?
- How could I tell whether or not my model output is accurate?

The answers to these questions will dictate how you build your eval, and what kind of data you need to assemble before you can evaluate model performance. Let's make this more concrete with a real example.

Example objective - sentiment analysis of movie reviews

Let's say you are building an LLM application to perform sentiment analysis on movie reviews from IMDB. You'd like to give the model the text of a movie review, and have it output `1` if it's a positive review, and `0` if it's a negative review. For this use case, let's answer the three questions posed above:

What kind of output do I want the model to generate?

In this case, you want the model to generate just a `1` for positive reviews, and `0` for negative reviews. You don't want the model to generate anything else besides these two numbers.

What kinds of inputs should the model be able to handle?

The model should be able to process user-generated movie review text. Based on the text of the reviews, the model should be able to ascertain whether or not the overall tone of the review is positive or negative. For testing purposes, we will assume that all the reviews being tested are in English.

How could I tell whether or not my model output is accurate?

There are a few potential ways we could tell if the model output is accurate:

- We could ask another (ideally larger or more powerful) model to analyze the same review. If the other model agreed with our model's output, that could be a strong indicator of success.
- We could compare the model output to human-created output on the same review, which we could trust to be accurate. Human-labeled data could be considered **ground truth**, which is the reality we want to test our model outputs against.

The second option is likely to be most accurate - so if we have the ability to test against human-labeled data as our ground truth, that's probably the best option. Model-graded outputs can be useful when human-labeled data is not available, though.

Now that we have a firm grasp on what we want the model to generate, what kinds of inputs we want to test with, and how we'd know if the ouputs are accurate, we can move on to collecting the data we need for our eval.

Assemble test data

Getting good test data inputs is arguably the hardest part of building evals. That's because good test data must be representative of the kinds of inputs the model will receive in production.

Consider our movie review use case. If all our test cases were negative reviews written by the same person, it would not be a very good test dataset. We'd ideally want the percentage of positive and negative reviews to match the proportions we observe in reality, and to be written by a diverse range of authors.

When generating a test dataset, you have a few options:

- You can find an existing open source dataset that matches your use case
- You can create a synthetic dataset, either by having a human or a model generate test inputs that meet your specifications
- You can create a dataset from real production usage of your application

The third option is ideal, since it is likely to resemble the reality your application will face. If you are already using OpenAI, you might consider using Stored Completions for your API traffic. Setting `store: true` on your completions will make them show up here in the dashboard, where you can filter them to create a dataset for evals, fine-tuning, or distillation.

Store completions in the API with metadata

javascript

```javascript
import OpenAI from "openai";
const openai = new OpenAI();

const response = await openai.chat.completions.create({
  model: "gpt-4o",
  messages: [
    { role: "system", content: "You are a corporate IT support expert." },
    { role: "user", content: "How can I hide the dock on my Mac?"},
  ],
  store: true,
  metadata: {
    role: "manager",
    department: "accounting",
    source: "homepage"
  }
});

console.log(response.choices[0]);
```

For the purposes of this guide, however, we'll be using an open source dataset from Hugging Face. You can download a CSV version of the data from here.

This dataset contains two columns - the `text` of the review, and a `label` that is either `0` or `1` depending on if the review is negative or positive, respectively. The data looks something like the following:

```
text,label
"I love sci-fi and am willing...",0
"Worth the entertainment value ...",1
"its a totally average film ...",0
```

In this case, the test data contains both the test inputs for the model, **and our ground truth labels** for the review content. Now that we have a dataset we can use for our eval, complete with ground truth labels, we can generate model output to test.

Generate model outputs

Next, you will want to define the prompt you'll use to generate model output, given the data inputs from your test dataset. The output of this prompt will be graded during your eval run for accuracy and quality.

For the movie review use case, we'll use a prompt that looks like this:

System message

```
You are an expert in analyzing the sentiment of movie reviews.

- If the movie review is positive, you will output a 1
- If the movie review is negative, you will output a 0
- Only output 1 or 0 as a response - do not output any other text
```

User message

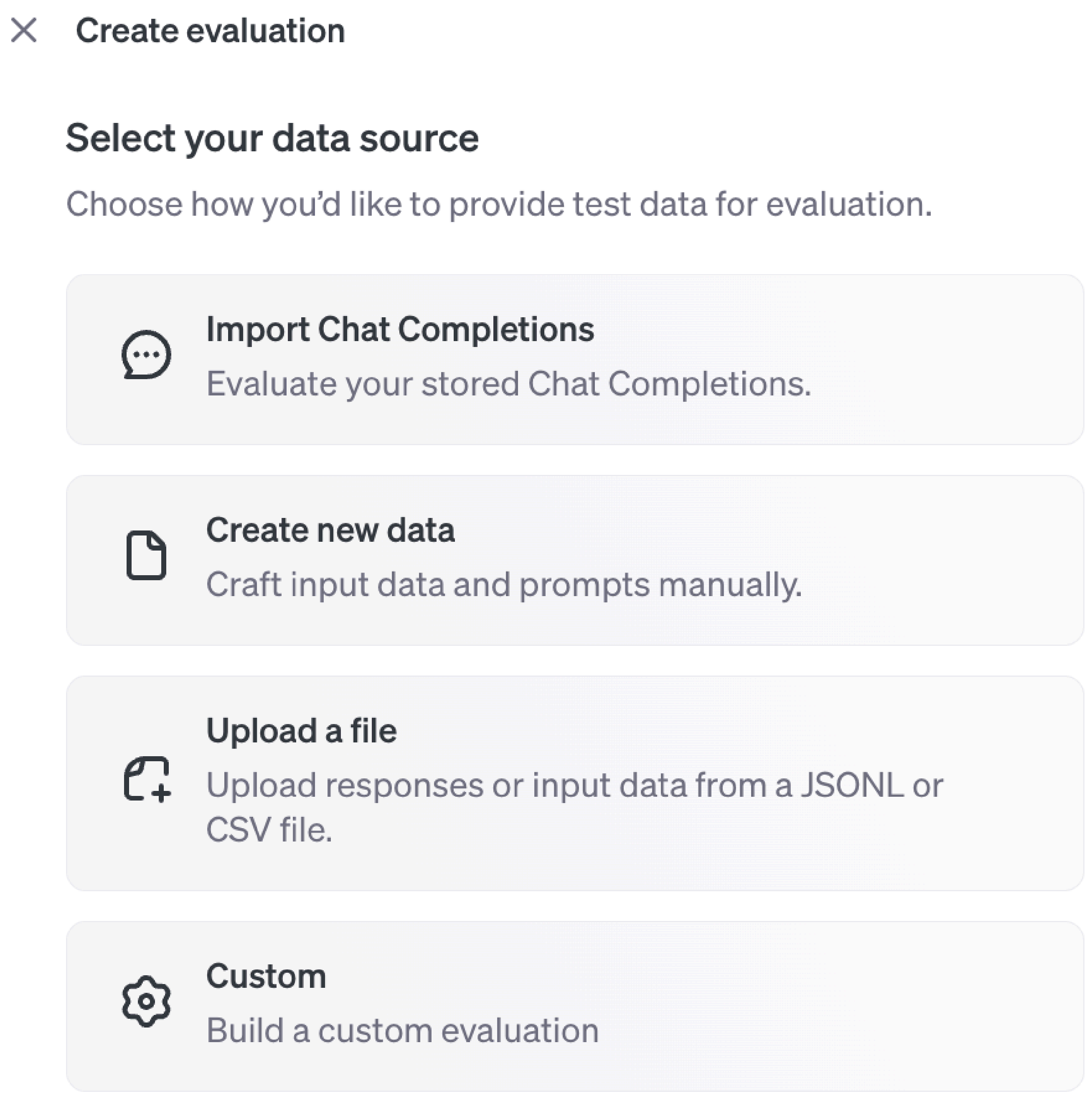
```
Determine if the following review is positive or negative:

<review text here>
```

Using the test data and the prompt as configured above, we are ready to configure and run an eval in the dashboard!

Set up and run your eval

With all of our ingredients assembled, we're ready to set up and run our eval in the OpenAI API dashboard. Go to the evals page and click **Create**. There are several options to help you set up an eval, but let's choose the **Custom** option to set everything up from scratch.



Create evaluation

Select your data source

Choose how you'd like to provide test data for evaluation.

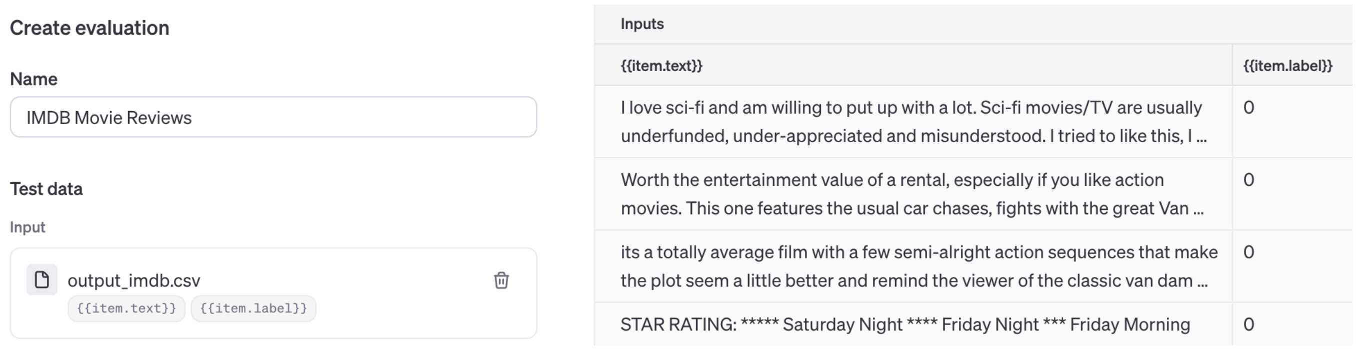
Import Chat Completions
Evaluate your stored Chat Completions.

Create new data
Craft input data and prompts manually.

Upload a file
Upload responses or input data from a JSONL or CSV file.

Custom
Build a custom evaluation

After your new custom eval is created, give the eval a name, and then upload the CSV version of IMDB movie review data from here. The data upload is found under the **Test data** header.



Create evaluation

Name: IMDB Movie Reviews

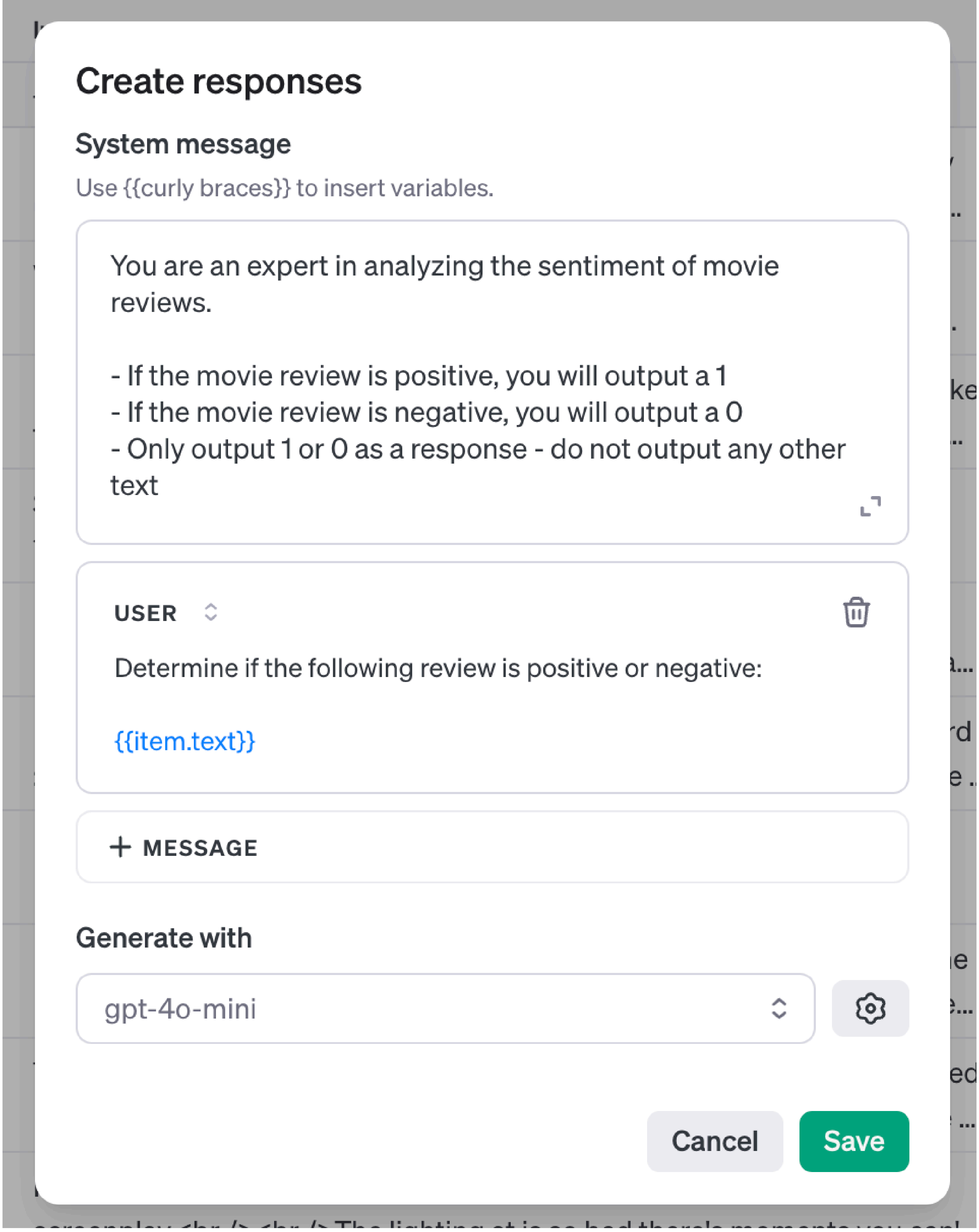
Test data

Input: output_imdb.csv `{{item.text}}` `{{item.label}}`

Inputs

{{item.text}}	{{item.label}}
I love sci-fi and am willing to put up with a lot. Sci-fi movies/TV are usually underfunded, under-appreciated and misunderstood. I tried to like this, I ...	0
Worth the entertainment value of a rental, especially if you like action movies. This one features the usual car chases, fights with the great Van ...	0
its a totally average film with a few semi-alright action sequences that make the plot seem a little better and remind the viewer of the classic van dam ...	0
STAR RATING: ***** Saturday Night **** Friday Night *** Friday Morning	0

Next, we'll configure the prompt we want to use to generate model outputs that we want to test. Under the **Responses** header, use the system and user messages shown above to generate model outputs with each test case. The `{{ item.text }}` expression will substitute in the value in the `text` column of your data file CSV.



Create responses

System message

Use {{curly braces}} to insert variables.

```
You are an expert in analyzing the sentiment of movie reviews.

- If the movie review is positive, you will output a 1
- If the movie review is negative, you will output a 0
- Only output 1 or 0 as a response - do not output any other text
```

USER

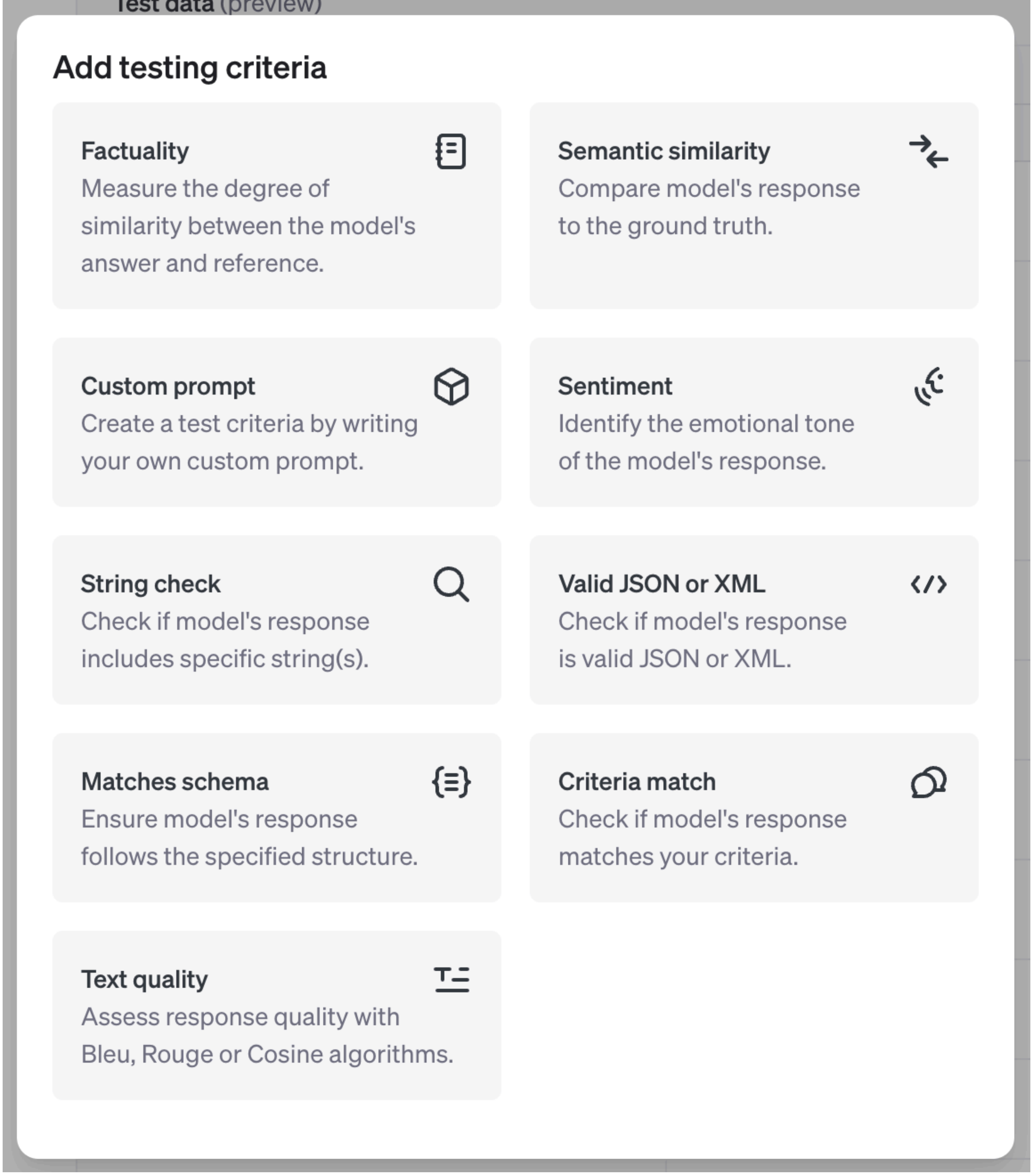
```
Determine if the following review is positive or negative:

{{item.text}}
```

Generate with: gpt-4o-mini

You can generate responses with any OpenAI model you'd like.

Finally, we need to configure the test criteria for the eval under the **Test criteria** header. These criteria will be used to determine whether or not our model output is what we expect.


Test data (preview)

Add testing criteria

Factuality
Measure the degree of similarity between the model's answer and reference.

Semantic similarity
Compare model's response to the ground truth.

Custom prompt
Create a test criteria by writing your own custom prompt.

Sentiment
Identify the emotional tone of the model's response.

String check
Check if model's response includes specific string(s).

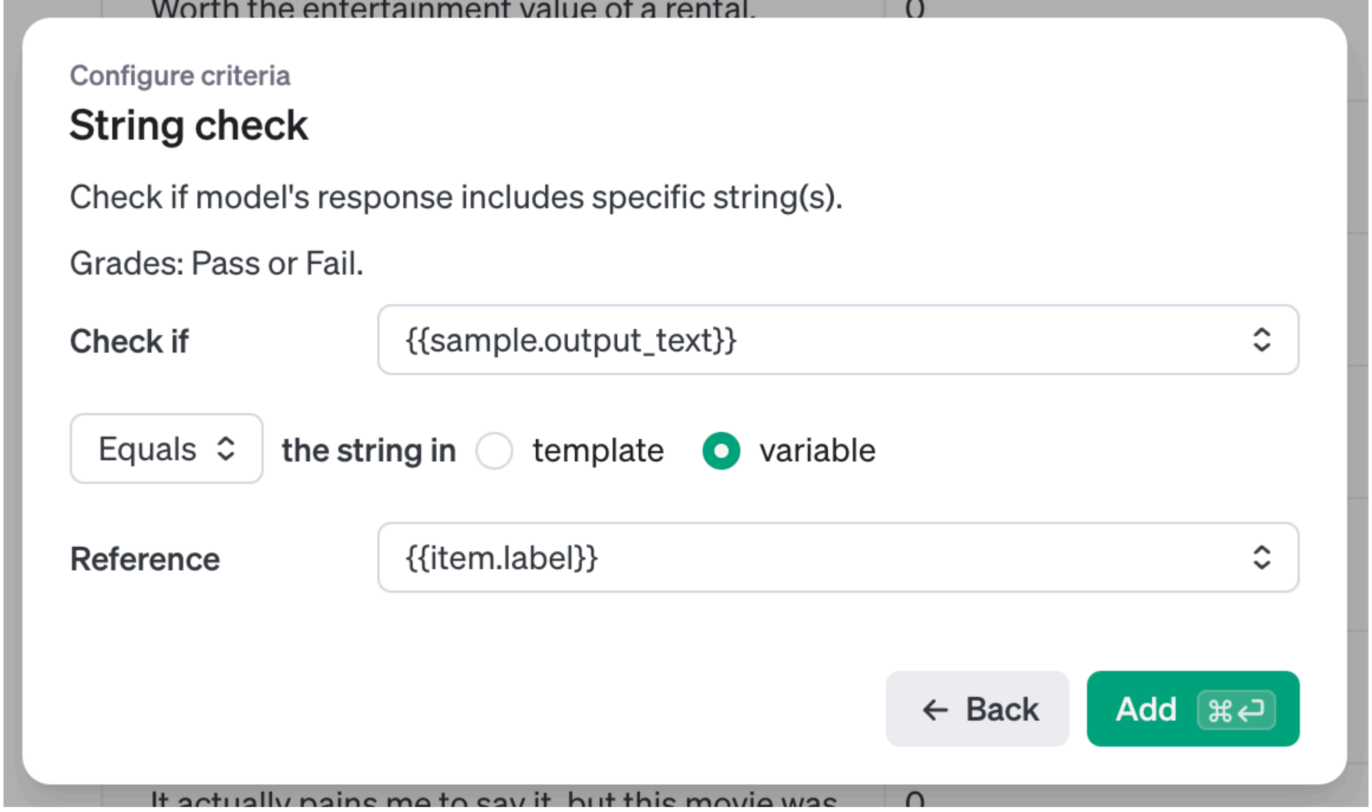
Valid JSON or XML
Check if model's response is valid JSON or XML.

Matches schema
Ensure model's response follows the specified structure.

Criteria match
Check if model's response matches your criteria.

Text quality
Assess response quality with Bleu, Rouge or Cosine algorithms.

There are several graders you can choose from, like assessing text quality or creating a custom model grader from a prompt. In this case, we want to test the output of the model against a ground truth output that is either `0` or `1`. For this kind of comparison, we can use the **String check** grader.


Configure criteria

String check

Check if model's response includes specific string(s).

Grades: Pass or Fail.

Check if: {{sample.output_text}}

Equals the string in ○ template [✓ variable]

Reference: {{item.label}}

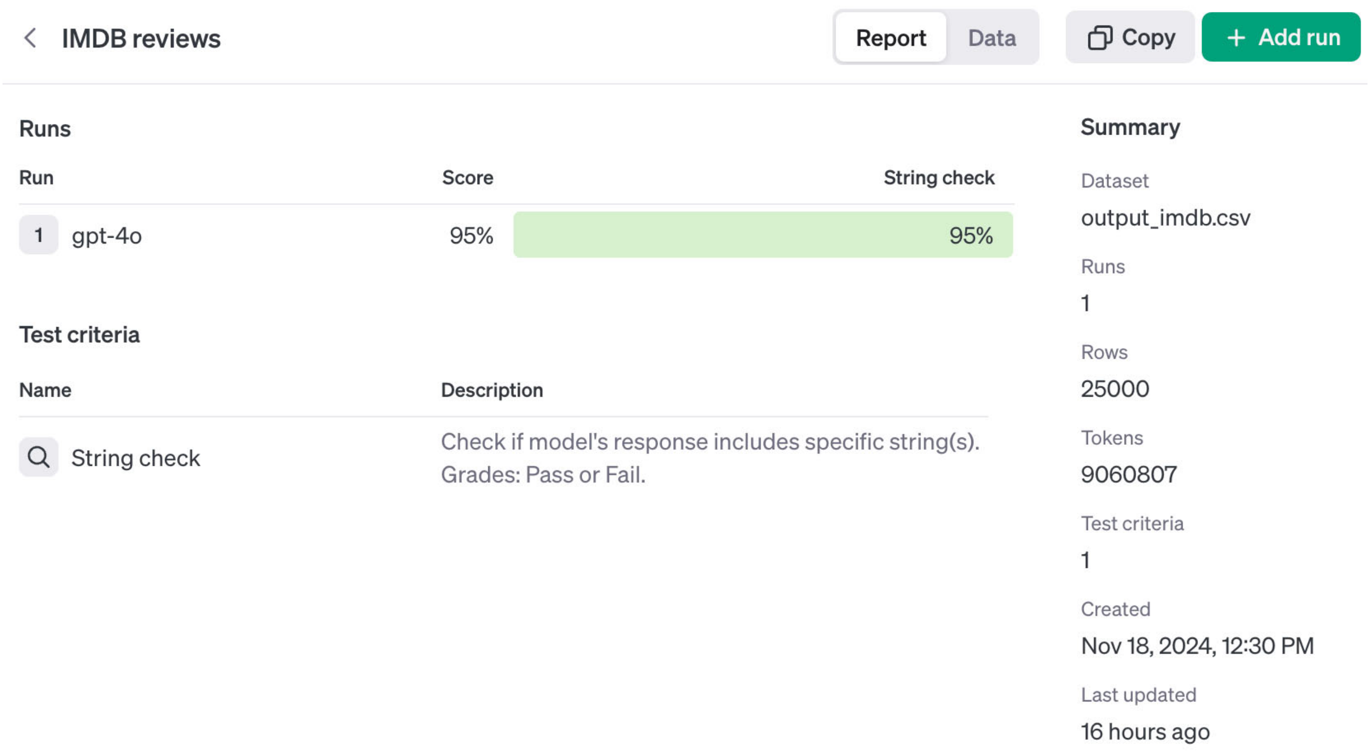
Worth the entertainment value of a rental.
0
← Back
Add ⌘↵
It actually pains me to say it, but this movie was
0

In this criteria, we are going to ensure that the model output matches our ground truth value.

- `{{ sample.output_text }}` - output from the model
- `{{ item.label }}` - ground truth label from the dataset

When your eval is successfully configured, you can press the **Run** button to run the eval!

Depending on the size of the dataset, your eval may run in the background for quite some time. It's okay to navigate away from the Evaluations page until it finishes. When the run completes, you can view the results in the dashboard:



Runs

Run	Score	String check
1 gpt-4o	95%	95%

Test criteria

Name	Description
String check	Check if model's response includes specific string(s). Grades: Pass or Fail.

Summary

Dataset	output_imdb.csv
Runs	1
Rows	25000
Tokens	9060807
Test criteria	1
Created	Nov 18, 2024, 12:30 PM
Last updated	16 hours ago

You can drill into the data to see which checks failed, and debug the results by either cleaning up your test data (perhaps some reviews were incorrectly labeled) or experiment with improvements to your prompt.

Before long, you will be an expert in determining whether or not your LLM applications are producing the outputs you expect.

Next steps

Now you know how to create and run evals in the OpenAI dashboard! Here are a few other resources that may be useful to you as you continue to improve your model results.

Fine-tuning
Improve a model's ability to generate responses tailored to your use case.

Model distillation
Learn how to distill large model results to smaller, cheaper, and faster models.