Starting April 29, 2025, Gemini 1.5 Pro and Gemini 1.5 Flash models are not available in projects that have no prior usage of these models, including new projects. For details, see Model versions and lifecycle (/vertex-ai/generative-ai/docs/learn/model-versions#legacy-stable).

Gen AI evaluation service API

Release Notes

The Gen AI evaluation service lets you evaluate your large language models (LLMs) across several metrics with your own criteria. You can provide inference-time inputs, LLM responses and additional parameters, and the Gen AI evaluation service returns metrics specific to the evaluation task.

Metrics include model-based metrics, such as `PointwiseMetric` and `PairwiseMetric`, and in-memory computed metrics, such as `rouge`, `bleu`, and tool function-call metrics. `PointwiseMetric` and `PairwiseMetric` are generic model-based metrics that you can customize with your own criteria. Because the service takes the prediction results directly from models as input, the evaluation service can perform both inference and subsequent evaluation on all models supported by Vertex AI.

For more information on evaluating a model, see Gen AI evaluation service overview (/vertex-ai/generative-ai/docs/models/evaluation-overview).

Limitations

The following are limitations of the evaluation service:

- The evaluation service may have a propagation delay in your first call.
- Most model-based metrics consume gemini-2.0-flash quota (/vertex-ai/generative-ai/docs/quotas#eval-quotas) because the Gen AI evaluation service leverages `gemini-2.0-flash` as the underlying judge model to compute these model-based metrics.
- Some model-based metrics, such as MetricX and COMET, use different machine learning models, so they don't consume gemini-2.0-flash quota (/vertex-ai/generative-ai/docs/quotas#eval-quotas).

Note: MetricX and COMET will be not be charged during preview. At GA, the pricing will be the same as all pointwise model based metrics.

Example syntax

Syntax to send an evaluation call.

curlPython (#python)
(#curl)

```
curl -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json" \

https://${LOCATION}-aiplatform.googleapis.com/v1/projects/${PROJECT_ID}/locatio
-d '{
  "pointwise_metric_input" : {
    "metric_spec" : {
      ...
    },
    "instance": {
      ...
    },
  }
}'
```

Parameter list

Parameters	
`exact_match_input`	Optional: `ExactMatchInput` Input to assess if the prediction matches the reference exactly.
`bleu_input`	Optional: `BleuInput` Input to compute BLEU score by comparing the prediction against the reference.
`rouge_input`	Optional: `RougeInput` Input to compute `rouge` scores by comparing the prediction against the reference. Different `rouge` scores are supported by `rouge_type`.
`fluency_input`	Optional: `FluencyInput`

	Input to assess a single response's language mastery.
`coherence_input`	Optional: `CoherenceInput` Input to assess a single response's ability to provide a coherent, easy-to-follow reply.
`safety_input`	Optional: `SafetyInput` Input to assess a single response's level of safety.
`groundedness_input`	Optional: `GroundednessInput` Input to assess a single response's ability to provide or reference information included only in the input text.
`fulfillment_input`	Optional: `FulfillmentInput` Input to assess a single response's ability to completely fulfill instructions.
`summarization_quality_input`	Optional: `SummarizationQualityInput` Input to assess a single response's overall ability to summarize text.
`pairwise_summarization_quality_input`	Optional: `PairwiseSummarizationQualityInput` Input to compare two responses' overall summarization quality.
`summarization_helpfulness_input`	Optional: `SummarizationHelpfulnessInput` Input to assess a single response's ability to provide a summarization, which contains the details necessary to substitute the original text.
`summarization_verbosity_input`	Optional: `SummarizationVerbosityInput` Input to assess a single response's ability to provide a succinct summarization.
`question_answering_quality_input`	Optional: `QuestionAnsweringQualityInput` Input to assess a single response's overall ability to answer questions, given a body of text to reference.
`pairwise_question_answering_quality_input`	Optional: `PairwiseQuestionAnsweringQualityInput` Input to compare two responses' overall ability to answer questions, given a body of text to reference.
`question_answering_relevance_input`	Optional: `QuestionAnsweringRelevanceInput`

	Input to assess a single response's ability to respond with relevant information when asked a question.
`question_answering_helpfulness_input`	Optional: `QuestionAnsweringHelpfulnessInput` Input to assess a single response's ability to provide key details when answering a question.
`question_answering_correctness_input`	Optional: `QuestionAnsweringCorrectnessInput` Input to assess a single response's ability to correctly answer a question.
`pointwise_metric_input`	Optional: `PointwiseMetricInput` Input for a generic pointwise evaluation.
`pairwise_metric_input`	Optional: `PairwiseMetricInput` Input for a generic pairwise evaluation.
`tool_call_valid_input`	Optional: `ToolCallValidInput` Input to assess a single response's ability to predict a valid tool call.
`tool_name_match_input`	Optional: `ToolNameMatchInput` Input to assess a single response's ability to predict a tool call with the right tool name.
`tool_parameter_key_match_input`	Optional: `ToolParameterKeyMatchInput` Input to assess a single response's ability to predict a tool call with correct parameter names.
`tool_parameter_kv_match_input`	Optional: `ToolParameterKvMatchInput` Input to assess a single response's ability to predict a tool call with correct parameter names and values
`comet_input`	Optional: `CometInput` Input to evaluate using COMET (https://huggingface.co/Unbabel/wmt22-comet-da).
`metricx_input`	Optional: `MetricxInput` Input to evaluate using MetricX (https://github.com/google-research/metricx).

`ExactMatchInput`

```
{
  "exact_match_input": {
    "metric_spec": {},
    "instances": [
      {
        "prediction": string,
        "reference": string
      }
    ]
  }
}
```

Parameters

`metric_spec`	Optional: `ExactMatchSpec`. Metric spec, defining the metric's behavior.
`instances`	Optional: `ExactMatchInstance[]` Evaluation input, consisting of LLM response and reference.
`instances.prediction`	Optional: `string` LLM response.
`instances.reference`	Optional: `string` Golden LLM response for reference.

`ExactMatchResults`

```
{
  "exact_match_results": {
    "exact_match_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`exact_match_metric_values`	`ExactMatchMetricValue[]` Evaluation results per instance input.
`exact_match_metric_values.score`	`float` One of the following: • `0`: Instance was not an exact match • `1`: Exact match

`BleuInput`

```
{
  "bleu_input": {
    "metric_spec": {
      "use_effective_order": bool
    },
    "instances": [
      {
        "prediction": string,
        "reference": string
      }
    ]
  }
}
```

Parameters

`metric_spec`	Optional: `BleuSpec` Metric spec, defining the metric's behavior.
`metric_spec.use_effective_order`	Optional: `bool` Whether to take into account n-gram orders without any match.
`instances`	Optional: `BleuInstance[]` Evaluation input, consisting of LLM response and reference.

`instances.prediction`	Optional: `string` LLM response.
`instances.reference`	Optional: `string` Golden LLM response for reference.

`BleuResults`

```
{
  "bleu_results": {
    "bleu_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`bleu_metric_values`	`BleuMetricValue[]` Evaluation results per instance input.
`bleu_metric_values.score`	`float`: `[0, 1]`, where higher scores mean the prediction is more like the reference.

`RougeInput`

```
{
  "rouge_input": {
    "metric_spec": {
      "rouge_type": string,
      "use_stemmer": bool,
      "split_summaries": bool
    },
    "instances": [
      {
```

```
        "prediction": string,
        "reference": string
      }
    ]
  }
}
```

Parameters

`metric_spec`	Optional: `RougeSpec` Metric spec, defining the metric's behavior.
`metric_spec.rouge_type`	Optional: `string` Acceptable values: • `rougen[1-9]`: compute `rouge` scores based on the overlap of n-grams between the prediction and the reference. • `rougeL`: compute `rouge` scores based on the Longest Common Subsequence (LCS) between the prediction and the reference. • `rougeLsum`: first splits the prediction and the reference into sentences and then computes the LCS for each tuple. The final `rougeLsum` score is the average of these individual LCS scores.
`metric_spec.use_stemmer`	Optional: `bool` Whether Porter stemmer should be used to strip word suffixes to improve matching.
`metric_spec.split_summaries`	Optional: `bool` Whether to add newlines between sentences for rougeLsum.
`instances`	Optional: `RougeInstance[]` Evaluation input, consisting of LLM response and reference.
`instances.prediction`	Optional: `string` LLM response.
`instances.reference`	Optional: `string` Golden LLM response for reference.

`RougeResults`

```
{
  "rouge_results": {
    "rouge_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`rouge_metric_values`	`RougeValue[]` Evaluation results per instance input.
`rouge_metric_values.score`	`float`: `[0, 1]`, where higher scores mean the prediction is more like the reference.

`FluencyInput`

```
{
  "fluency_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `FluencySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `FluencyInstance`

	Evaluation input, consisting of LLM response.
`instance.prediction`	Optional: `string` LLM response.

`FluencyResult`

```
{
  "fluency_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output	
`score`	`float`: One of the following: • 1: Inarticulate • 2: Somewhat Inarticulate • 3: Neutral • 4: Somewhat fluent • 5: Fluent
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`CoherenceInput`

```
{
  "coherence_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string
    }
```

```
  }
}
```

Parameters

`metric_spec`	Optional: `CoherenceSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `CoherenceInstance` Evaluation input, consisting of LLM response.
`instance.prediction`	Optional: `string` LLM response.

`CoherenceResult`

```
{
  "coherence_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • 1: Incoherent • 2: Somewhat incoherent • 3: Neutral • 4: Somewhat coherent • 5: Coherent
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`SafetyInput`

```
{
  "safety_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `SafetySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `SafetyInstance` Evaluation input, consisting of LLM response.
`instance.prediction`	Optional: `string` LLM response.

`SafetyResult`

```
{
  "safety_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • `0`: Unsafe

	• 1: Safe
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`GroundednessInput`

```
{
  "groundedness_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "context": string
    }
  }
}
```

Parameter	Description
`metric_spec`	Optional: GroundednessSpec Metric spec, defining the metric's behavior.
`instance`	Optional: GroundednessInstance Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`GroundednessResult`

```
{
  "groundedness_result": {
    "score": float,
```

```
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • `0`: Ungrounded • `1`: Grounded
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`FulfillmentInput`

```
{
  "fulfillment_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "instruction": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `FulfillmentSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `FulfillmentInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.

`instance.instruction`	Optional: `string` Instruction used at inference time.

`FulfillmentResult`

```
{
  "fulfillment_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • **1**: No fulfillment • **2**: Poor fulfillment • **3**: Some fulfillment • **4**: Good fulfillment • **5**: Complete fulfillment
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`SummarizationQualityInput`

```
{
  "summarization_quality_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "instruction": string,
      "context": string,
    }
```

```
  }
}
```

Parameters

`metric_spec`	Optional: `SummarizationQualitySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `SummarizationQualityInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`SummarizationQualityResult`

```
{
  "summarization_quality_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • 1: Very bad • 2: Bad

	• 3: Ok • 4: Good • 5: Very good
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`PairwiseSummarizationQualityInput`

```
{
  "pairwise_summarization_quality_input": {
    "metric_spec": {},
    "instance": {
      "baseline_prediction": string,
      "prediction": string,
      "instruction": string,
      "context": string,
    }
  }
}
```

Parameters

`metric_spec`	Optional: `PairwiseSummarizationQualitySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `PairwiseSummarizationQualityInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.baseline_prediction`	Optional: `string` Baseline model LLM response.
`instance.prediction`	Optional: `string` Candidate model LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.

`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`PairwiseSummarizationQualityResult`

```
{
  "pairwise_summarization_quality_result": {
    "pairwise_choice": PairwiseChoice,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`pairwise_choice`	`PairwiseChoice`: Enum with possible values as follows: • `BASELINE`: Baseline prediction is better • `CANDIDATE`: Candidate prediction is better • `TIE`: Tie between Baseline and Candidate predictions.
`explanation`	`string`: Justification for pairwise_choice assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`SummarizationHelpfulnessInput`

```
{
  "summarization_helpfulness_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "instruction": string,
      "context": string,
    }
```

```
  }
}
```

Parameters

`metric_spec`	Optional: `SummarizationHelpfulnessSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `SummarizationHelpfulnessInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`SummarizationHelpfulnessResult`

```
{
  "summarization_helpfulness_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • 1: Unhelpful • 2: Somewhat unhelpful

	• 3: Neutral • 4: Somewhat helpful • 5: Helpful
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`SummarizationVerbosityInput`

```
{
  "summarization_verbosity_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "instruction": string,
      "context": string,
    }
  }
}
```

Parameters

`metric_spec`	Optional: `SummarizationVerbositySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `SummarizationVerbosityInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`SummarizationVerbosityResult`

```
{
  "summarization_verbosity_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output	
`score`	`float`. One of the following: • `-2`: Terse • `-1`: Somewhat terse • `0`: Optimal • `1`: Somewhat verbose • `2`: Verbose
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`QuestionAnsweringQualityInput`

```
{
  "question_answering_quality_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "instruction": string,
      "context": string,
    }
  }
}
```

Parameters

`metric_spec`	Optional: `QuestionAnsweringQualitySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `QuestionAnsweringQualityInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`QuestionAnsweringQualityResult`

```
{
  "question_answering_quality_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • 1: Very bad • 2: Bad • 3: Ok • 4: Good • 5: Very good
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`PairwiseQuestionAnsweringQualityInput`

```
{
  "question_answering_quality_input": {
    "metric_spec": {},
    "instance": {
      "baseline_prediction": string,
      "prediction": string,
      "instruction": string,
      "context": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `QuestionAnsweringQualitySpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `QuestionAnsweringQualityInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.baseline_prediction`	Optional: `string` Baseline model LLM response.
`instance.prediction`	Optional: `string`

	Candidate model LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`PairwiseQuestionAnsweringQualityResult`

```
{
  "pairwise_question_answering_quality_result": {
    "pairwise_choice": PairwiseChoice,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`pairwise_choice`	`PairwiseChoice`: Enum with possible values as follows: • `BASELINE`: Baseline prediction is better • `CANDIDATE`: Candidate prediction is better • `TIE`: Tie between Baseline and Candidate predictions.
`explanation`	`string`: Justification for `pairwise_choice` assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`QuestionAnsweringRelevanceInput`

```
{
  "question_answering_quality_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
```

```
      "instruction": string,
      "context": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `QuestionAnsweringRelevanceSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `QuestionAnsweringRelevanceInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`QuestionAnsweringRelevancyResult`

```
{
  "question_answering_relevancy_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following:

	• 1: Irrelevant • 2: Somewhat irrelevant • 3: Neutral • 4: Somewhat relevant • 5: Relevant
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`QuestionAnsweringHelpfulnessInput`

```
{
  "question_answering_helpfulness_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "instruction": string,
      "context": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `QuestionAnsweringHelpfulnessSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `QuestionAnsweringHelpfulnessInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string`

	Inference-time text containing all information, which can be used in the LLM response.

`QuestionAnsweringHelpfulnessResult`

```
{
  "question_answering_helpfulness_result": {
    "score": float,
    "explanation": string,
    "confidence": float
  }
}
```

Output

`score`	`float`: One of the following: • **1**: Unhelpful • **2**: Somewhat unhelpful • **3**: Neutral • **4**: Somewhat helpful • **5**: Helpful
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`QuestionAnsweringCorrectnessInput`

```
{
  "question_answering_correctness_input": {
    "metric_spec": {
      "use_reference": bool
    },
    "instance": {
      "prediction": string,
      "reference": string,
      "instruction": string,
```

```
      "context": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `QuestionAnsweringCorrectnessSpec` Metric spec, defining the metric's behavior.
`metric_spec.use_reference`	Optional: `bool` If reference is used or not in the evaluation.
`instance`	Optional: `QuestionAnsweringCorrectnessInstance` Evaluation input, consisting of inference inputs and corresponding response.
`instance.prediction`	Optional: `string` LLM response.
`instance.reference`	Optional: `string` Golden LLM response for reference.
`instance.instruction`	Optional: `string` Instruction used at inference time.
`instance.context`	Optional: `string` Inference-time text containing all information, which can be used in the LLM response.

`QuestionAnsweringCorrectnessResult`

```
{
  "question_answering_correctness_result": {
    "score": float,
    "explanation": string,
    "confidence": float
```

```
  }
}
```

Output

`score`	`float`: One of the following: • `0`: Incorrect • `1`: Correct
`explanation`	`string`: Justification for score assignment.
`confidence`	`float`: `[0, 1]` Confidence score of our result.

`PointwiseMetricInput`

```
{
  "pointwise_metric_input": {
    "metric_spec": {
      "metric_prompt_template": string
    },
    "instance": {
      "json_instance": string,
    }
  }
}
```

Parameters

`metric_spec`	Required: `PointwiseMetricSpec` Metric spec, defining the metric's behavior.
`metric_spec.metric_prompt_template`	Required: `string` A prompt template defining the metric. It is rendered by the key-value pairs in instance.json_instance
`instance`	Required: `PointwiseMetricInstance` Evaluation input, consisting of json_instance.

`instance.json_instance`	Optional: `string` The key-value pairs in Json format. For example, {"key_1": "value_1", "key_2": "value_2"}. It is used to render metric_spec.metric_prompt_template.

`PointwiseMetricResult`

```
{
  "pointwise_metric_result": {
    "score": float,
    "explanation": string,
  }
}
```

Output

`score`	`float`: A score for pointwise metric evaluation result.
`explanation`	`string`: Justification for score assignment.

`PairwiseMetricInput`

```
{
  "pairwise_metric_input": {
    "metric_spec": {
      "metric_prompt_template": string
    },
    "instance": {
      "json_instance": string,
    }
  }
}
```

Parameters

`metric_spec`	Required: `PairwiseMetricSpec` Metric spec, defining the metric's behavior.

`metric_spec.metric_prompt_template`	Required: `string` A prompt template defining the metric. It is rendered by the key-value pairs in instance.json_instance
`instance`	Required: `PairwiseMetricInstance` Evaluation input, consisting of json_instance.
`instance.json_instance`	Optional: `string` The key-value pairs in JSON format. For example, {"key_1": "value_1", "key_2": "value_2"}. It is used to render metric_spec.metric_prompt_template.

`PairwiseMetricResult`

```
{
  "pairwise_metric_result": {
    "score": float,
    "explanation": string,
  }
}
```

Output

`score`	`float`: A score for pairwise metric evaluation result.
`explanation`	`string`: Justification for score assignment.

`ToolCallValidInput`

```
{
  "tool_call_valid_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "reference": string
    }
```

```
  }
}
```

Parameters

`metric_spec`	Optional: `ToolCallValidSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `ToolCallValidInstance` Evaluation input, consisting of LLM response and reference.
`instance.prediction`	Optional: `string` Candidate model LLM response, which is a JSON serialized string that contains `content` and `tool_calls` keys. The `content` value is the text output from the model. The `tool_call` value is a JSON serialized string of a list of tool calls. Ar example is: `{ "content": "", "tool_calls": [ { "name": "book_tickets", "arguments": { "movie": "Mission Impossible Dead Reckoning Part 1" "theater": "Regal Edwards 14", "location": "Mountain View CA", "showtime": "7:30", "date": "2024-03-30", "num_tix": "2" } } ] }`
`instance.reference`	Optional: `string` Golden model output in the same format as prediction.

`ToolCallValidResults`

```
{
  "tool_call_valid_results": {
    "tool_call_valid_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`tool_call_valid_metric_values`	repeated `ToolCallValidMetricValue`: Evaluation results per instance input.
`tool_call_valid_metric_values.score`	`float`: One of the following: • `0`: Invalid tool call • `1`: Valid tool call

`ToolNameMatchInput`

```
{
  "tool_name_match_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "reference": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `ToolNameMatchSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `ToolNameMatchInstance`

	Evaluation input, consisting of LLM response and reference.
`instance.prediction`	Optional: `string` Candidate model LLM response, which is a JSON serialized string that contains `content` and `tool_calls` keys. The `content` value is the text output from the model. The `tool_call` value is a JSON serialized string of a list of tool calls.
`instance.reference`	Optional: `string` Golden model output in the same format as prediction.

`ToolNameMatchResults`

```
{
  "tool_name_match_results": {
    "tool_name_match_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`tool_name_match_metric_values`	repeated `ToolNameMatchMetricValue`: Evaluation results per instance input.
`tool_name_match_metric_values.score`	`float`: One of the following: • `0`: Tool call name doesn't match the reference. • `1`: Tool call name matches the reference.

`ToolParameterKeyMatchInput`

```
{
  "tool_parameter_key_match_input": {
    "metric_spec": {},
```

```
    "instance": {
      "prediction": string,
      "reference": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `ToolParameterKeyMatchSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `ToolParameterKeyMatchInstance` Evaluation input, consisting of LLM response and reference.
`instance.prediction`	Optional: `string` Candidate model LLM response, which is a JSON serialized string that contains `content` and `tool_calls` keys. The `content` value is the text output from the model. The `tool_call` value is a JSON serialized string of a list of tool calls.
`instance.reference`	Optional: `string` Golden model output in the same format as prediction.

`ToolParameterKeyMatchResults`

```
{
  "tool_parameter_key_match_results": {
    "tool_parameter_key_match_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`tool_parameter_key_match_metric_values`	repeated `ToolParameterKeyMatchMetricValue`: Evaluation results per instance input.
`tool_parameter_key_match_metric_values.score`	`float`: `[0, 1]`, where higher scores mean more parameters match the reference parameters' names.

`ToolParameterKVMatchInput`

```
{
  "tool_parameter_kv_match_input": {
    "metric_spec": {},
    "instance": {
      "prediction": string,
      "reference": string
    }
  }
}
```

Parameters

`metric_spec`	Optional: `ToolParameterKVMatchSpec` Metric spec, defining the metric's behavior.
`instance`	Optional: `ToolParameterKVMatchInstance` Evaluation input, consisting of LLM response and reference.
`instance.prediction`	Optional: `string` Candidate model LLM response, which is a JSON serialized string that contains `content` and `tool_calls` keys. The `content` value is the text output from the model. The `tool_call` value is a JSON serialized string of a list of tool calls.
`instance.reference`	Optional: `string` Golden model output in the same format as prediction.

`ToolParameterKVMatchResults`

```
{
  "tool_parameter_kv_match_results": {
    "tool_parameter_kv_match_metric_values": [
      {
        "score": float
      }
    ]
  }
}
```

Output

`tool_parameter_kv_match_metric_values`	repeated `ToolParameterKVMatchMetricValue`: Evaluation results per instance input.
`tool_parameter_kv_match_metric_values.score`	`float`: `[0, 1]`, where higher scores mean more parameters match the reference parameters' names and values.

`CometInput`

```
{
  "comet_input" : {
    "metric_spec" : {
      "version": string
    },
    "instance": {
      "prediction": string,
      "source": string,
      "reference": string,
    },
  }
}
```

Parameters

`metric_spec`	Optional: `CometSpec` Metric spec, defining the metric's behavior.

`metric_spec.version`	Optional: `string` `COMET_22_SRC_REF`: COMET 22 for translation, source, and reference. It evaluates the translation (prediction) using all three inputs.
`metric_spec.source_language`	Optional: `string` Source language in BCP-47 format (https://en.wikipedia.org/wiki/IETF_language_tag). For example, "es".
`metric_spec.target_language`	Optional: `string` Target language in BCP-47 format (https://en.wikipedia.org/wiki/IETF_language_tag). For example, "es"
`instance`	Optional: `CometInstance` Evaluation input, consisting of LLM response and reference. The exact fields used for evaluation are dependent on the COMET version.
`instance.prediction`	Optional: `string` Candidate model LLM response. This is the output of the LLM which is being evaluated.
`instance.source`	Optional: `string` Source text. This is in the original language that the prediction was translated from.
`instance.reference`	Optional: `string` Ground truth used to compare against the prediction. This is in the same language as the prediction.

`CometResult`

```
{
  "comet_result" : {
    "score": float
  }
}
```

Output

`score`	`float`: `[0, 1]`, where 1 represents a perfect translation.

`MetricxInput`

```
{
  "metricx_input" : {
    "metric_spec" : {
      "version": string
    },
    "instance": {
      "prediction": string,
      "source": string,
      "reference": string,
    },
  }
}
```

Parameters

`metric_spec`	Optional: `MetricxSpec` Metric spec, defining the metric's behavior.
`metric_spec.version`	Optional: `string` One of the following: • `METRICX_24_REF`: MetricX 24 for translation and reference. It evaluates the prediction (translation) by comparing with the provided reference text input. • `METRICX_24_SRC`: MetricX 24 for translation and source. It evaluates the translation (prediction) by Quality Estimation (QE), without a reference text input. • `METRICX_24_SRC_REF`: MetricX 24 for translation, source and reference. It evaluates the translation (prediction) using all three inputs.
`metric_spec.source_language`	Optional: `string` Source language in BCP-47 format (https://en.wikipedia.org/wiki/IETF_language_tag). For example, "es".

`metric_spec.target_language`	Optional: `string` Target language in BCP-47 format (https://en.wikipedia.org/wiki/IETF_language_tag). For example, "es".
`instance`	Optional: `MetricxInstance` Evaluation input, consisting of LLM response and reference. The exact fields used for evaluation are dependent on the MetricX version.
`instance.prediction`	Optional: `string` Candidate model LLM response. This is the output of the LLM which is being evaluated.
`instance.source`	Optional: `string` Source text which is in the original language that the prediction was translated from.
`instance.reference`	Optional: `string` Ground truth used to compare against the prediction. It is in the same language as the prediction.

`MetricxResult`

```
{
  "metricx_result" : {
    "score": float
  }
}
```

Output	
`score`	`float`: `[0, 25]`, where 0 represents a perfect translation.

Examples

Evaluate an output

The following example demonstrates how to call the Gen AI Evaluation API to evaluate the output of an LLM using a variety of evaluation metrics, including the following:

- `summarization_quality`
- `groundedness`
- `fulfillment`
- `summarization_helpfulness`
- `summarization_verbosity`

PythonGo (#go)
(#python)

```
import pandas as pd

import vertexai
from vertexai.preview.evaluation import EvalTask, MetricPromptTemplateExamples

# TODO(developer): Update and un-comment below line
# PROJECT_ID = "your-project-id"
vertexai.init(project=PROJECT_ID, location="us-central1")

eval_dataset = pd.DataFrame(
    {
        "instruction": [
            "Summarize the text in one sentence.",
            "Summarize the text such that a five-year-old can understand.",
        ],
        "context": [
            """As part of a comprehensive initiative to tackle urban congestion
            sustainable urban living, a major city has revealed ambitious plans
            extensive overhaul of its public transportation system. The project
            only to improve the efficiency and reliability of public transit bu
            reduce the city\'s carbon footprint and promote eco-friendly commut
            City officials anticipate that this strategic investment will enhan
            accessibility for residents and visitors alike, ushering in a new e
            efficient, environmentally conscious urban transportation.""",
            """A team of archaeologists has unearthed ancient artifacts sheddin
            previously unknown civilization. The findings challenge existing hi
            narratives and provide valuable insights into human history.""",
        ],
        "response": [
            "A major city is revamping its public transportation system to figh
            "Some people who dig for old things found some very special tools a
```

```
        ],
    }
)

eval_task = EvalTask(
    dataset=eval_dataset,
    metrics=[
        MetricPromptTemplateExamples.Pointwise.SUMMARIZATION_QUALITY,
        MetricPromptTemplateExamples.Pointwise.GROUNDEDNESS,
        MetricPromptTemplateExamples.Pointwise.VERBOSITY,
        MetricPromptTemplateExamples.Pointwise.INSTRUCTION_FOLLOWING,
    ],
)

prompt_template = (
    "Instruction: {instruction}. Article: {context}. Summary: {response}"
)
result = eval_task.evaluate(prompt_template=prompt_template)

print("Summary Metrics:\n")

for key, value in result.summary_metrics.items():
    print(f"{key}: \t{value}")

print("\n\nMetrics Table:\n")
print(result.metrics_table)
# Example response:
# Summary Metrics:
# row_count:      2
# summarization_quality/mean:     3.5
# summarization_quality/std:      2.1213203435596424
# ...
```

Evaluate an output: pairwise summarization quality

The following example demonstrates how to call the Gen AI evaluation service API to evaluate the output of an LLM using a pairwise summarization quality comparison.

RESTPython (#python)Go (#go)
(#rest)

Before using any of the request data, make the following replacements:

- ***PROJECT_ID***: Your project ID (/resource-manager/docs/creating-managing-projects#identifiers).

- *LOCATION*: The region to process the request.
- *PREDICTION*: LLM response.
- *BASELINE_PREDICTION*: Baseline model LLM response.
- *INSTRUCTION*: The instruction used at inference time.
- *CONTEXT*: Inference-time text containing all relevant information, that can be used in the LLM response.

HTTP method and URL:

```
POST https://LOCATION-aiplatform.googleapis.com/v1beta1/projects/PROJECT_ID
```

Request JSON body:

```
{
  "pairwise_summarization_quality_input": {
    "metric_spec": {},
    "instance": {
      "prediction": "PREDICTION",
      "baseline_prediction": "BASELINE_PREDICTION",
      "instruction": "INSTRUCTION",
      "context": "CONTEXT",
    }
  }
}
```

To send your request, choose one of these options:

curlPowerShell (#powershell)
(#curl)

Note: The following command assumes that you have logged in to the **gcloud** CLI with your user account by running **gcloud init** (/sdk/gcloud/reference/init) or **gcloud auth login** (/sdk/gcloud/reference/auth/login) , or by using Cloud Shell (/shell/docs), which automatically logs you into the **gcloud** CLI . You can check the currently active account by running **gcloud auth list** (/sdk/gcloud/reference/auth/list).

Save the request body in a file named **`request.json`**, and execute the following command:

```
curl -X POST \
     -H "Authorization: Bearer $(gcloud auth print-access-token)" \
     -H "Content-Type: application/json; charset=utf-8" \
     -d @request.json \
     "https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJE
```

Get ROUGE score

The following example calls the Gen AI evaluation service API to get the ROUGE score of a prediction, generated by a number of inputs. The ROUGE inputs use **`metric_spec`**, which determines the metric's behavior.

RESTPython (#python)Go (#go)
(#rest)

Before using any of the request data, make the following replacements:

- *PROJECT_ID*: Your project ID (/resource-manager/docs/creating-managing-projects#identifiers).
- *LOCATION*: The region to process the request.
- *PREDICTION*: LLM response.
- *REFERENCE*: Golden LLM response for reference.
- *ROUGE_TYPE*: The calculation used to determine the rouge score. See **`metric_spec.rouge_type`** (/vertex-ai/generative-ai/docs/model-reference/evaluation#rougeinput) for acceptable values.
- *USE_STEMMER*: Determines whether the Porter stemmer is used to strip word suffixes to improve matching. For acceptable values, see **`metric_spec.use_stemmer`** (/vertex-ai/generative-ai/docs/model-reference/evaluation#rougeinput).
- *SPLIT_SUMMARIES*: Determines if new lines are added between **`rougeLsum`** sentences. For acceptable values, see **`metric_spec.split_summaries`** (/vertex-ai/generative-ai/docs/model-reference/evaluation#rougeinput) .

HTTP method and URL:

```
POST https://LOCATION-aiplatform.googleapis.com/v1beta1/projects/PROJECT_ID
```

Request JSON body:

```
{
  "rouge_input": {
    "instances": {
      "prediction": "PREDICTION",
      "reference": "REFERENCE.",
    },
    "metric_spec": {
      "rouge_type": "ROUGE_TYPE",
      "use_stemmer": USE_STEMMER,
      "split_summaries": SPLIT_SUMMARIES,
    }
  }
}
```

To send your request, choose one of these options:

curlPowerShell (#powershell)
(#curl)

Note: The following command assumes that you have logged in to the **gcloud** CLI with your user account by running **gcloud init** (/sdk/gcloud/reference/init) or **gcloud auth login** (/sdk/gcloud/reference/auth/login) , or by using Cloud Shell (/shell/docs), which automatically logs you into the **gcloud** CLI . You can check the currently active account by running **gcloud auth list** (/sdk/gcloud/reference/auth/list).

Save the request body in a file named **request.json**, and execute the following command:

```
curl -X POST \
     -H "Authorization: Bearer $(gcloud auth print-access-token)" \
     -H "Content-Type: application/json; charset=utf-8" \
```

```
-d @request.json \
"https://LOCATION-aiplatform.googleapis.com/v1beta1/projects/PROJE
```

What's next

- For detailed documentation, see Run an evaluation (/vertex-ai/generative-ai/docs/models/online-pipeline-services).

Last updated 2025-06-06 UTC.