TRACE — Tracking RAG Accuracy and Consistency Evaluation

(FastAPI Endpoint)

1. Overview

The **TRACE Endpoint** evaluates the performance and reliability of Retrieval-Augmented Generation (RAG) systems by running selected **TRACE metrics** (powered by Ragas) on user-specified data.

The endpoint accepts a **string** `query` describing:

- The **metrics** to evaluate
  (e.g., `"answer_accuracy"`, `"context_recall"`, `"context_utilisation"`, etc.)
- The **input parameters** for those metrics (question, contexts, predicted answer, ground truth, etc.)

Flow:

1. **Parse**: Convert incoming `query` (JSON string or natural language) into a canonical JSON payload.
2. **Validate**: Check if all required parameters for requested TRACE metrics are present.
3. **Flag Missing**: If any parameters are missing for a test, set `missing: true` for that test and return the missing list.
4. **Evaluate**: If all parameters are available, compute the evaluation score for each metric.
5. **Respond**: Return JSON with provided parameters, requested tests, missing parameter info, and evaluation scores.

Output JSON contains:

- `provided_parameters` — parsed parameters received
- `tests` — TRACE metrics requested
- `missing` — per-metric missing parameters or `false` if none missing
- `evaluation_scores` — scores per metric (0–1 range)
- `details` — extra computation details per metric

2. Endpoint specification

HTTP:
`POST /trace`

Request body:

```
{ "query": "<string>" }
```

Where `query` is either:

- A **JSON string** matching TRACE schema, or
- Natural-language text (agent attempts to parse).

Example `query` **(JSON string)**:

```
{ "tests": ["answer_accuracy","context_recall","context_utilisation"], "question":
"What is the capital of France?", "contexts": [ {"id":"ctx1","text":"Paris is the
capital of France."}, {"id":"ctx2","text":"Lyon is in France."} ], "answer": "Paris",
"ground_truth": "Paris", "relevant_context_ids": ["ctx1"] }
```

Example response (success):

```
{ "provided_parameters": { ... }, "tests":
["answer_accuracy","context_recall","context_utilisation"], "missing": {
"answer_accuracy": false, "context_recall": false, "context_utilisation": false },
"evaluation_scores": { "answer_accuracy": 1.0, "context_recall": 1.0,
"context_utilisation": 0.86 }, "details": { "context_recall": { "relevant_context_ids":
["ctx1"], "found_relevant_count": 1, "total_relevant_count": 1 },
"context_utilisation": { "overlap_tokens": 3, "context_tokens_total": 10 } } }
```

3. TRACE Canonical Payload Schema

After parsing, TRACE works on this JSON format:

```
{ "tests": ["answer_accuracy", "context_recall", ...], "question": "<string>",
"contexts": [{"id":"<id>", "text":"<text>"}], "answer": "<string>", "ground_truth": "
<string>|[string]", "relevant_context_ids": ["id1","id2"], "extra": {} }
```

4. Supported TRACE metrics

Metric	Required Parameters	Core Idea
answer_accuracy	`answer` , `ground_truth`	Compares predicted answer with ground truth (exact/semantic).
context_recall	`contexts` , `relevant_context_ids`	Fraction of relevant contexts present in provided contexts.
context_precision	`contexts` , `relevant_context_ids`	Fraction of provided contexts that are relevant.
faithfulness	`answer` , `contexts` (optional: `ground_truth`)	Checks if the answer is supported by provided contexts.
answer_relevancy	`question` , `answer`	Measures how well the answer addresses the question.

Metric	Required Parameters	Core Idea
context_utilisation	`question` , `contexts` , `answer`	Measures how much of the provided context was actually used in the answer.

5. TRACE Agent Behavior

1. **Parse Query** → JSON format.
2. **Validate Parameters** → Check against metric requirements.
3. **If Missing** → Return missing parameters, skip scoring.
4. **If Complete** → Run metric evaluation functions (Ragas-powered).
5. **Return** → Provided params + scores + details.

6. Example FastAPI Implementation — TRACE

```
@app.post("/trace") async def trace_evaluation(raw: RawQuery): # 1) Parse query string
into canonical payload try: payload = parse_query_to_json(raw.query) except ValueError
as e: raise HTTPException(status_code=400, detail=str(e)) # 2) Validate parameters
missing = {} for test in payload.tests: reqs = METRIC_REQUIREMENTS.get(test) if not
reqs: missing[test] = ["unsupported_test"] continue miss = [r for r in reqs if
getattr(payload, r, None) in (None, [], "")] missing[test] = miss # Early return if
missing if any(missing.values()): return { "provided_parameters":
json.loads(payload.json()), "tests": payload.tests, "missing": missing,
"evaluation_scores": {}, "details": {} } # 3) Compute metrics scores, details = {}, {}
# ... metric computation logic here ... return { "provided_parameters":
json.loads(payload.json()), "tests": payload.tests, "missing": missing,
"evaluation_scores": scores, "details": details }
```

7. Example TRACE Input & Output

Input:

```
{ "query": "{\"tests\":[\"context_recall\"],\"question\":\"Who wrote 'Pride and
Prejudice'?\",\"contexts\":[{\"id\":\"c1\",\"text\":\"Jane Austen authored Pride and
Prejudice.\"}],\"answer\":\"Jane Austen\",\"ground_truth\":\"Jane
Austen\",\"relevant_context_ids\":[\"c1\"]}" }
```

Output:

```
{ "provided_parameters": { ... }, "tests": ["context_recall"], "missing": {
"context_recall": false }, "evaluation_scores": { "context_recall": 1.0 }, "details": {
```

```
"context_recall": { "relevant_context_ids": ["c1"], "found_relevant_count": 1,
"total_relevant_count": 1 } } }
```

8. Summary

- **TRACE** = Tracking RAG Accuracy and Consistency Evaluation.
- Provides a unified endpoint to validate RAG outputs against quality metrics.
- Returns missing parameter info if inputs are incomplete.
- Runs 6 key metrics to measure retrieval quality, answer correctness, and evidence usage.