Evaluation Framework for Knowledge Base

- Introduction
- Process
- Metrics
- Run tests
  - Run a test case
  - Run a test suite
  - Testing report
- Discussion
- Executing Tests
  - Create a Test Suite
  - Execute a Test Suite
  - Update 1:
  - Update 2:

Introduction

To evaluate the Knowledge Base (KB), we will implement a simple evaluation framework, based on DeepEval framework and Python package. The Knowledge Base is an implementation of a Retrieval Augmented Generation (RAG) system, and we will use the specific metrics for a RAG.

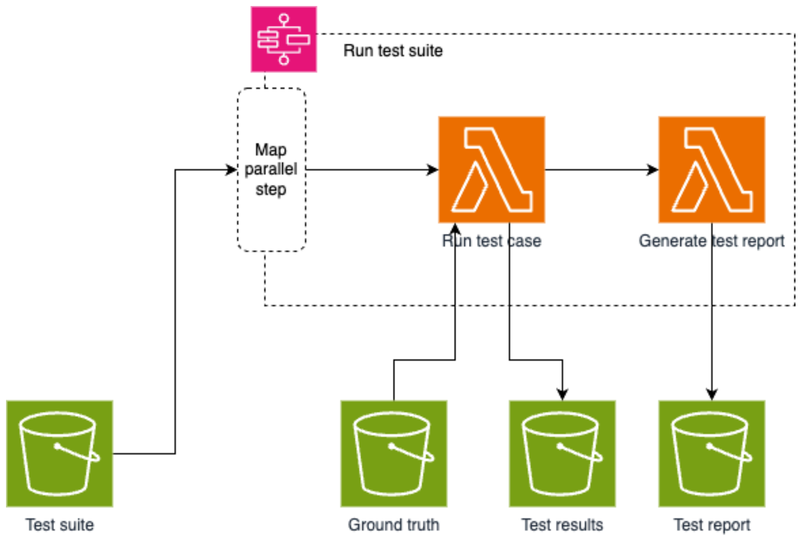
Note: this document refers to the prototype for the evaluation framework, see: LGA-87: [TIMEBOX 2 days] Prototype RAG system evaluation framework DONE

The parameters we consider are the following:

- **input** - this is the user prompt or the initial query
- **actual_output** - what the KB returns as an answer to our query
- **expected_output** - what we expect that the KB will return as an answer to our query (ground truth)
- **retrieval_context** - the context retrieved from the KB vector database based on which the KB Agent will formulate the actual output (the answer).

These parameters are used to calculate the metrics (see below).

Process



Note: the above approximative architecture diagram role is for understanding the process, should not be interpreted as prescriptive for the actual components to be included in the actual implementation.

Metrics

The following table summarises the metrics, with their description, as well as the resources needed to generate each metric, and the parameters needed.

Metric	Description	Resources needed	Parameters needed
Rouge Metric	**ROUGE**, which is short for **Recall-Oriented Understudy for Gisting Evaluation**, is a set of metrics used to evaluate how similar a generated text is to a reference text. It measures how much overlap there is between the model's output and a reference answer.	scorer	actual_output expected_output
Bert Score	Measures a LLM. Leverages the pre-trained contextual embeddings from **BERT** and matches tokens in candidate and reference sentences by cosine similarity. It computes the precision, recall, and F1 score. Note: We are only retaining and output F1 score in our implementation.	BERT model & tokenizer	actual_output expected_output
Faithfullness	Measures the quality of **RAG retriever**/LLM by evaluating whether the actual_output factually aligns with retrieved_context - it is a self-explaining LLM evaluation	Claude model	input actual_output retrieval_context
Answer Relevancy	Measures the quality of **RAG retriever** by evaluating how relevant the actual_output of the application is	Claude model	input actual_output

	compared with the provided input - it is a self-explaining LLM evaluation		
Contextual Precision	Measures the **RAG retriever** by evaluating whether nodes in retrieval_context that are relevant to the input are ranked higher than irrelevant ones - it is a self-explaining LLM evaluation	Claude model	input actual_output expected_output retrieval_context
Contextual Recall	Measures the quality of **RAG retriever** by evaluating the extent of which the retrieval_context aligns with the expected_output - it is a self-explaining LLM evaluation	Claude model	input actual_output expected_output retrieval_context
Contextual Relevancy	Measures the relevance of **RAG retriever** by evaluating the overall relevance of the information presented in your retrieval_context for a given input - it is a self-explaining LLM evaluation	Claude model	input actual_output retrieval_context
Bias	Determines whether LLM output contains gender, racial, or political bias. This can occur after fine-tuning a custom model from any RLHF or optimization. It is a referenceless metric	Claude model	input actual_output
Toxicity	Evaluates toxicity in LLM output. Particularly useful for a fine-tuning case. It is a referenceless metric.	Claude model	input actual_output

Run tests

Run a test case

To run a test case, will run the metrics ensemble against the definition of a test_case.

```
# Test case definition
test_case = LLMTestCase(
    input= input_data,
    actual_output=actual_output_data,
    expected_output=expected_output_data,
    retrieval_context=retrieval_context_data
)

# Metrics evaluation
metrics = [
    RougeMetric(),
    BERTScoreMetric(),
    FaithfulnessMetric(model=aws_bedrock),
    AnswerRelevancyMetric(model=aws_bedrock),
    ContextualPrecisionMetric(model=aws_bedrock),
    ContextualRecallMetric(model=aws_bedrock),
    ContextualRelevancyMetric(model=aws_bedrock),
    BiasMetric(model=aws_bedrock),
    ToxicityMetric(model=aws_bedrock)
]
# Run evaluation on the test case
for metric in metrics:
    metric.measure(test_case)
```

Run a test suite

To run a test suite, we run repeatedly (in parallel) the lambda that run a test case. Each run will save the metric measure / test case on S3.

Testing report

The testing report contains:

- the result for each test case, with this structure:

```
{
    "metric": metric.__name__,
    "score": metric.score,
    "success": metric.success,
    "reason": metric.reason
}
```

- the aggregated report for entire test suite, here is an example:

	name	Average Score	Pass rate
0	Rouge Metric	0.461070	0.4
1	BERT Score	0.571103	0.0
2	Faithfulness	1.000000	1.0
3	Answer Relevancy	1.000000	1.0
4	Contextual Precision	0.916667	1.0
5	Contextual Recall	1.000000	1.0
6	Contextual Relevancy	0.800000	1.0
7	Bias	0.000000	1.0
8	Toxicity	0.000000	1.0

Discussion

The impact of poorly formulated expected_output will be primarily on:

- Rouge Metric, BERT Score - very small score, 0 or very small pass rate.
- Contextual recall, Contextual precision - reduced score

Executing Tests

Create a Test Suite

1. Use the following template to create your test suite, placing your list of questions and expected answers in the relevant columns. test-suite-template.xlsx
2. Upload your file to the TestRunnerBucket in the test-suites folder.

Execute a Test Suite

The test runner is a step function which can run a test suite against either another step function, a lambda or a specific bedrock agent alias.

1. Create a new execution of the TestRunnerStateMachine
2. provide an event.json which follows the format

```
{
  "key": "test-suites/YOUR_TEST_SUITE_FILE.XLSX",
  "target": {
  // either
    "agentId":"AGENT_ID_HERE",
    "agentAliasId":"AGENT_ALIAS_ID",

  // or
    "stepFunctionArn":"STEP_FUNCTION_ARN_HERE",

  // or
    "lambdaArn": "LAMBDA_ARN_HERE"
  }
}

/*
 if you update the target object to include a "metric": "NAME OF METRIC" it will evaluate that metric specifically

 here is the list of names to use (exact match)
 Rouge Metric
 BERT Score
 Bias
 Toxicity
 Faithfulness
 Answer Relevancy
 Contextual Precision
 Contextual Recall
 Contextual Relevancy
*/

// examples

{
  "key": "test-suites/dp-test-suite.xlsx",
  "target": {
    "agentId": "E8NYLN6J5L",
    "agentAliasId": "IL55MJT2ZB",
  }
}
```

```

{
  "key": "test-suites/dp-test-suite.xlsx",
  "target": {
    "stepFunctionArn": "arn:aws:states:eu-west-2:420498525515:stateMachine:CombinedRagFusionChatRagFusionChatStateMachine7FADB45B-TcTnLbpEm5Ns"
  }
}

{
  "key": "test-suites/dp-test-suite.xlsx",
  "target": {
    "lambdaArn": "arn:aws:lambda:eu-west-2:420498525515:function:lhasa-genai-development-s-RagFusionChatBaseAgent9B-d9bh3SWThLdv"
  }
}

{
  "key": "test-suites/dp-test-suite-2.xlsx",
  "target": {
    "stepFunctionArn": "arn:aws:states:eu-west-2:420498525515:stateMachine:CombinedRagFusionChatRagFusionChatStateMachine7FADB45B-TcTnLbpEm5Ns",
    "metric": "BERT Score"
  }
}

```

   a. when using an agent the question will be passed to the agent.
   b. when using a lambda or step function arn the target will be provided with a json object with a `question` property `{ "question" : "your test suite question here"}`

3. The runner will execute the test suite, generate a report and provide the result to the step function in json. Additionally it creates two csv reports the first `details.csv` containing the details and evaluation metrics of every question and a second `result.csv` containing the aggregated result of the full test suite execution. these are stored in the TestRunnerBucket using the following using the target and timestamp of the execution in the path format:
   a. **Agents** - /test-reports/**agentId**-**agentAliasId/yymmdd-hhmmss**-[details/result].csv
   b. **Step Function**- /test-reports/**stepFunctionName/yymmdd-hhmmss**-[details/result].csv
   c. **Lambda** - /test-reports/**lambdaName/yymmdd-hhmmss**-[details/result].csv

If the target provides a response the following metrics will be evaluated:

- Rouge
- BERTScore
- Bias
- Toxicity

If the target responds with the citations used, it will also evaluate:

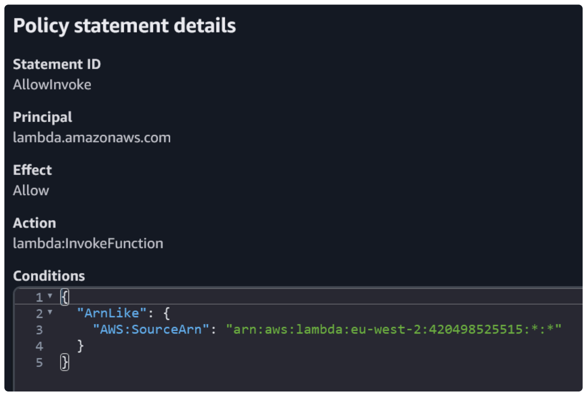
- Faithfulness
- Answer Relevancy
- Contextual Precision
- Contextual Recall
- Contextual Relevancy

There is an opportunity to include the following metrics once multi-turn history is implemented

- Conversation Relevancy

- Conversation Completeness

when targeting a lambdaArn you may encounter **An error occurred (AccessDeniedException) when calling the Invoke operation**. To resolve this you must add a resource based policy statement to the target lambda function allowing invocation by the test runner state machine. here is an example of the policy



Policy statement details

Statement ID
AllowInvoke

Principal
lambda.amazonaws.com

Effect
Allow

Action
lambda:InvokeFunction

Conditions

```
{
  "ArnLike": {
    "AWS:SourceArn": "arn:aws:lambda:eu-west-2:420498525515:*:*"
  }
}
```

Lastly, the test runner internally parses the responses of lambdas and stepfunctions differently and therefore we should standardise response jsons from these services so that the test runner can be simplified.

Update 1:

Now you can now specify a target metric so that only that metric is evaluated. Use one of the following metric names:

- Rouge Metric
- BERT Score
- Bias
- Toxicity
- Faithfulness
- Answer Relevancy
- Contextual Precision
- Contextual Recall
- Contextual Relevancy

like so:

```
{
  "key": "test-suites/dp-test-suite-2.xlsx",
  "target": {
    "stepFunctionArn": "arn:aws:states:eu-west-2:420498525515:stateMachine:CombinedRagFusionChatRagFusionChatStateMachine7FADB45B-TcTnLbpEm5Ns",
    "metric": "BERT Score"
  }
}
```

Update 2:

Now you can specify the number of iterations to have the test runner run the test suite. The test runner will run for each iteration, the full response generation and evaluation flow for each question in the test suite

Specify the number of iterations like so:

```
{
  "key": "test-suites/dp-test-suite-2.xlsx",
  "iterations": 2,
  "target": {
    "stepFunctionArn": "arn:aws:states:eu-west-2:420498525515:stateMachine:CombinedRagFusionChatRagFusionChatStateMachine7FADB45B-TcTnLbpEm5Ns",
    "metric": "BERT Score"
  }
}
```

IMPORTANT:
the execution input “iterations” property ...

1. must contain a positive integer value
2. is always required and does not have a default value
3. the execution will fail if missing from the input