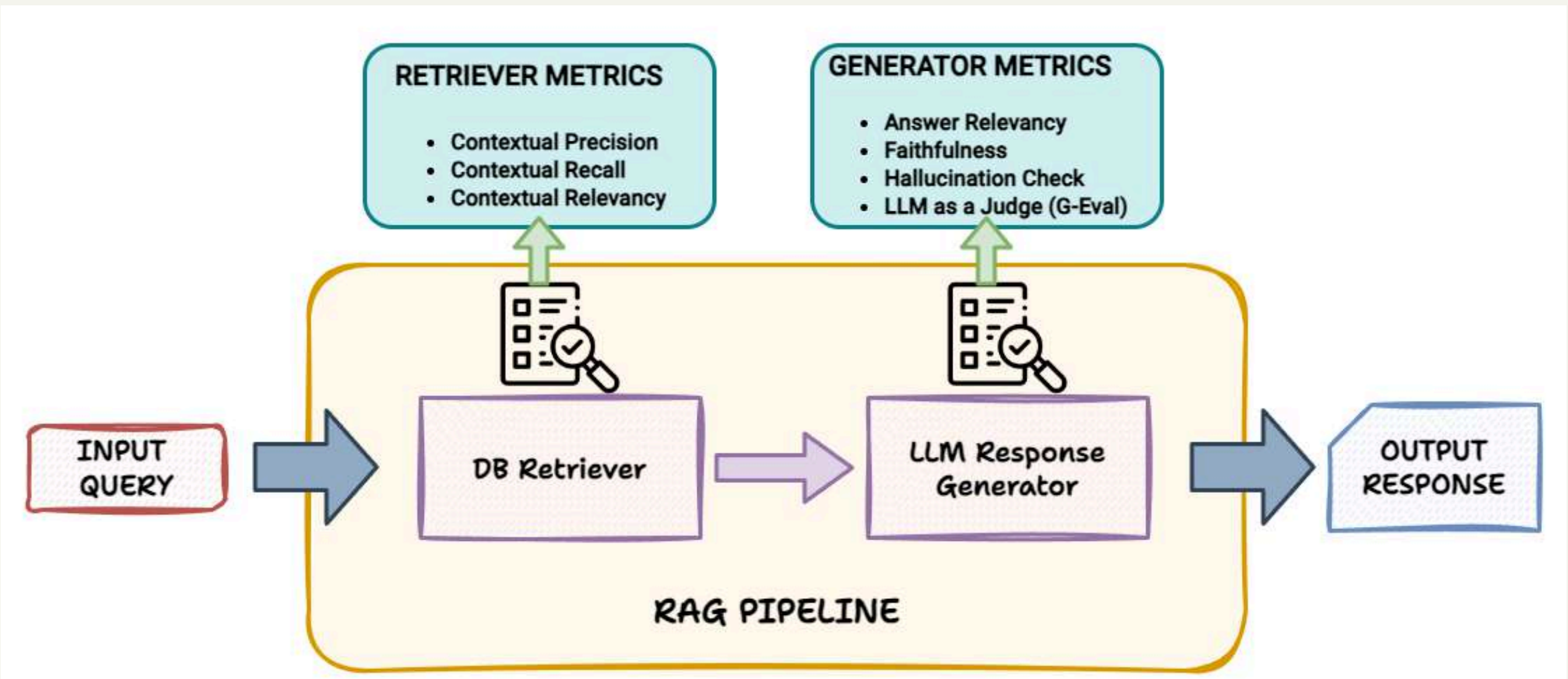
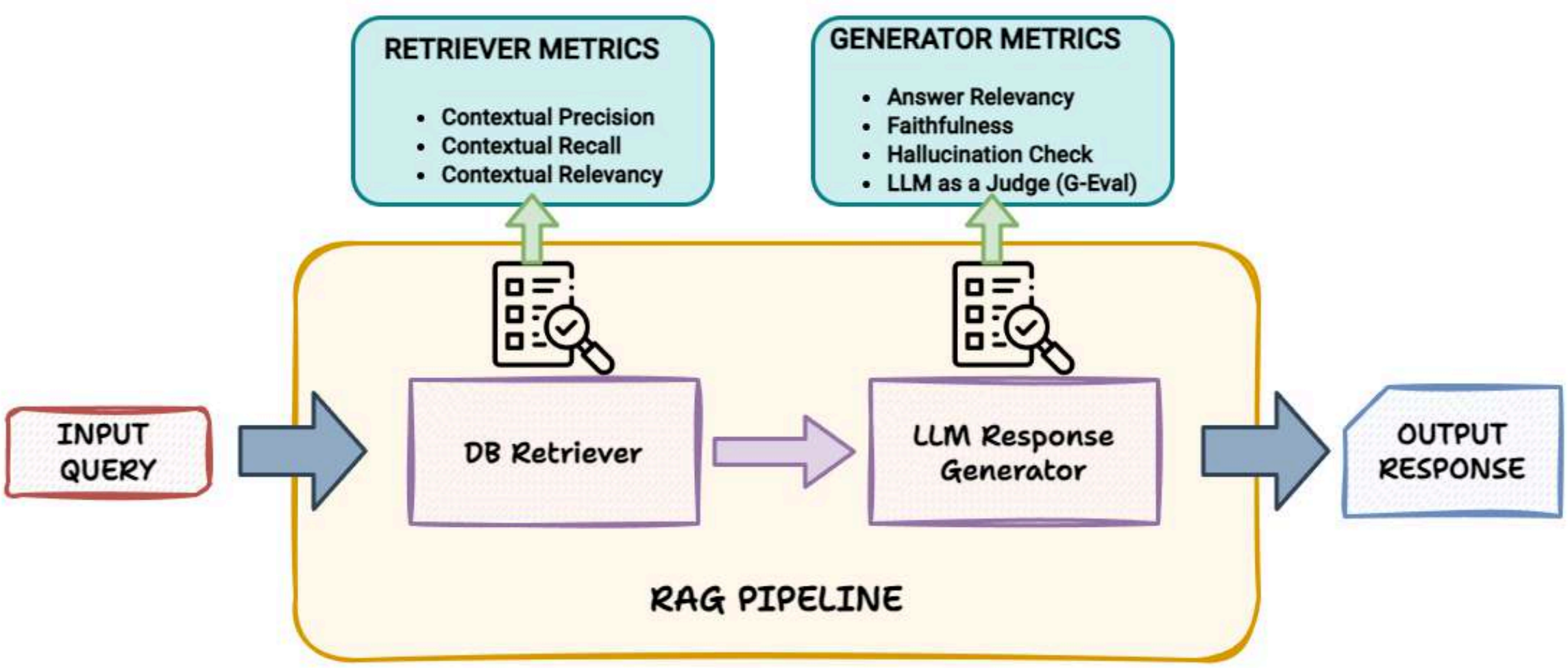
Comprehensive Guide to LLM & RAG System Evaluation Metrics



Dipanjan (DJ)

Standard RAG Evaluation Metrics



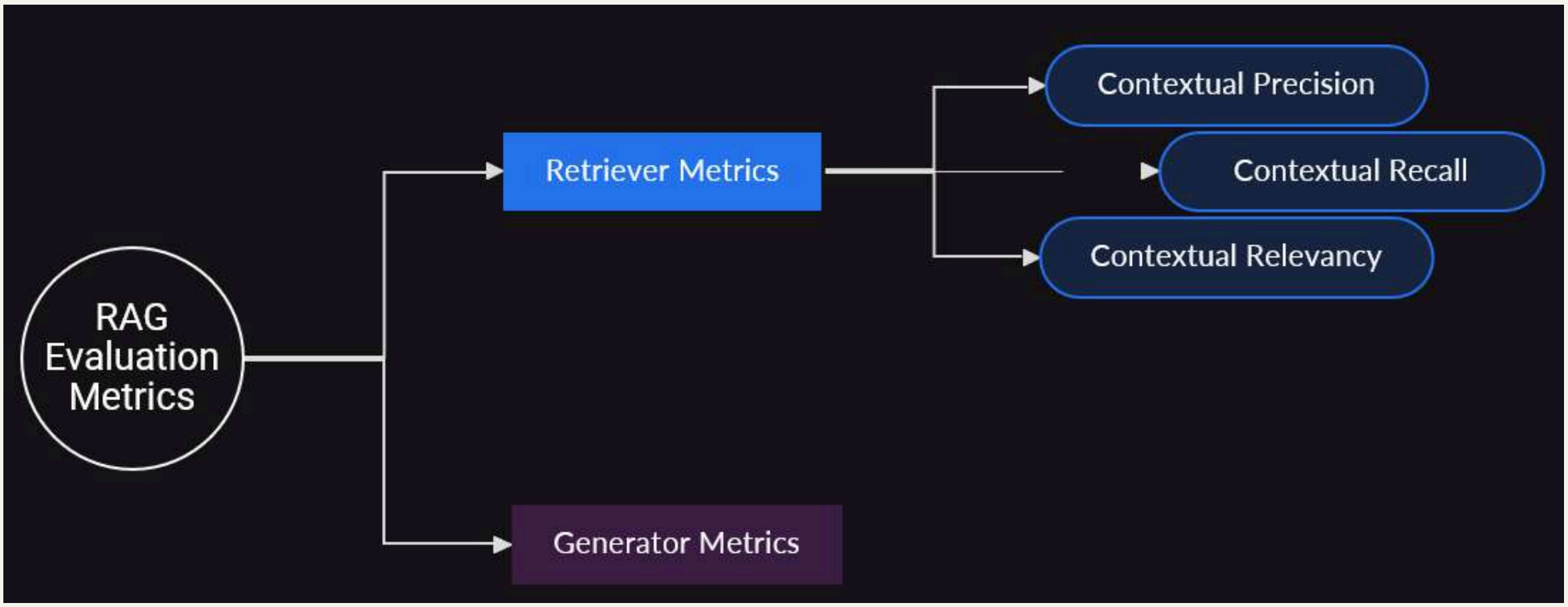
Two major points in a RAG System need evaluation:

- **Retriever:** This is where we measure retrieval performance from the Vector DB for input queries
  - **Contextual Precision:** Relevant retrieved context to input query should rank higher
  - **Contextual Recall:** Retrieved context should align with expected ground truth response
  - **Contextual Relevancy:** Relevancy of statements in retrieved context to the input query should be more in count

- **Generator:** This is where we measure the quality of generated responses from the LLM for input queries and retrieved context
  - **Answer Relevancy:** Relevancy of statements in generated response to the input query should be more in count or semantically similar (LLM-based or semantic similarity)
  - **Faithfulness:** Count of truthful claims made in the generated response w.r.t the retrieved context should be more
  - **Hallucination Check:** Number of statements in generated response which contradict the ground truth context should be minimal
  - **Custom LLM as a Judge:** You can create your own judging metrics based on custom evaluation criteria as needed

Source: Dipanjan (DJ)

Retriever Metrics



- **Context Precision:** Measures, whether **retrieved context** document chunks (nodes) that are relevant to the given **input query**, are ranked higher than irrelevant ones

- **Context Recall:** Measures the extent of which of the **retrieved context** document chunks (nodes) aligns with the **expected response answer** (ground truth reference)

- **Context Relevancy:** Measures the relevancy of the information in the **retrieved context** document chunks (nodes) to the given **input query**

Source: Dipanjan (DJ)

Contextual Precision

$$\text{Contextual Precision} = \frac{1}{\text{Number of Relevant Nodes}} \sum_{k=1}^{n} \left(\frac{\text{Number of Relevant Nodes Up to Position } k}{k} \times r_k \right)$$

Explanation of Key Variables:

- **Number of Relevant Nodes:** Total count of nodes that are considered relevant in the retrieval context.
- **k:** The position of the node in the retrieval context (starting from 1).
- **n:** Total length of the retrieval context.
- **Number of Relevant Nodes Up to Position k:** Cumulative count of relevant nodes up to and including position k.
- **r_k:** Binary relevance for the k^{th} node, where:
  - $r_k = 1$ if the node is relevant
  - $r_k = 0$ if the node is not relevant

- **Measures whether retrieved context document chunks (nodes) that are relevant to the given input query are ranked higher than irrelevant ones**

- **Higher Context Precision score represents a better retrieval system which can correctly rank relevant nodes higher**

Source: Dipanjan (DJ)

Contextual Precision

Example:

- Input Variables
  - "What is AI?"
- Retrieved Context:
  - **Node 1:** "Machine Learning is the study of algorithms which learn with more data."
  - **Node 2:** "AI is known as Artificial Intelligence."
  - **Node 3:** "Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning. AI includes applications like virtual assistants, robotics, and autonomous vehicles. It's evolving rapidly with advancements in machine learning and deep learning."
  - **Node 4:** "NLP is a branch of AI that enables computers to understand, interpret, and generate human language. Techniques include tokenization, stemming, and sentiment analysis. Applications range from chatbots to language translation services."
  - **Node 5:** "Machine learning is a field of artificial intelligence focused on enabling systems to learn patterns from data. Algorithms analyze past data to make predictions or classify information. Popular applications include recommendation systems and image recognition."

Metric Computation:

$$\text{Contextual Precision} = \frac{1}{\text{Number of Relevant Nodes}} \sum_{k=1}^{n} \left(\frac{\text{Number of Relevant Nodes Up to Position } k}{k} \times r_k \right)$$

Determine Relevance:

- Node 1: Not Relevant (r_k = 0)
- Node 2: Relevant (r_k = 1)
- Node 3: Relevant (r_k = 1)
- Node 4: Not Relevant (r_k = 0)
- Node 5: Not Relevant (r_k = 0)

Calculate Terms for Each Position k:

- For k = 1; r_k = 0
  - Term = 0 × 0 = 0
- For k = 2; r_k = 1
  - Number of Relevant Nodes Up to Position 2 = 1
  - Term = 1/2 × 1 = 0.5
- For k = 3; r_k = 1
  - Number of Relevant Nodes Up to Position 3 = 2
  - Term = 2/3 × 1 = 0.6667
- For k = 4; r/k = 0
  - Term = 2/4 × 0 = 0
- For k = 5; r/k = 0
  - Term = 2/5 × 0 = 0

Sum of Terms:

- 0 + 0.5 + 0.6667 + 0 + 0 = 1.1667

Final Calculation:

- Contextual Precision = 1.1667/2 = 0.5833

Source: Dipanjan (DJ)

Contextual Recall

$$\text{Contextual Recall} = \frac{\text{Number of Attributable Statements}}{\text{Total Number of Statements}}$$

Explanation of Key Variables:

- **Number of Attributable Statements:**
  - The count of statements in the expected output that can be attributed to nodes in the retrieved context.
- **Total Number of Statements:**
  - The total count of statements in the expected output.

How it Works:

- **Attributable Statements:**
  - Using an LLM as a judge, it identifies statements in the expected output that can be supported by information within the retrieved context.
- **Expected Output:**
  - The ideal or ground truth answer for the input query. This is used instead of the actual output to assess the retrieval system's performance quality.

- **Measures the extent of which of the retrieved context document chunks (nodes) aligns with the expected response answer (ground truth reference)**
- **Higher Context Recall score represents a better retrieval system which can capture all relevant context information from your Vector DB**

Source: Dipanjan (DJ)

Contextual Recall

Example:

1 Input Query:

- "What is AI?"

2 Expected Output:

- "AI, also known as Artificial Intelligence, is used to build complex systems for applications like virtual assistants, robotics, and autonomous vehicles."

3 Retrieved Context:

- Node 1: "NVIDIA makes chips for AI."
- Node 2: "AI is an acronym for Artificial Intelligence."

Metric Computation:

Identify Statements in the Expected Output:

- **Expected Output:** "AI, also known as Artificial Intelligence, is used to build complex systems for applications like virtual assistants, robotics, and autonomous vehicles."
- This output contains 2 statements:
  - **Statement 1:** "AI, also known as Artificial Intelligence..."
  - **Statement 2:** "...is used to build complex systems for applications like virtual assistants, robotics, and autonomous vehicles."

Assess Attributability for Each Statement:

- **Statement 1:** "AI, also known as Artificial Intelligence..."
  - This can be attributed to **Node 2** in the retrieved context: "AI is an acronym for Artificial Intelligence."
  - **Verdict:** Yes (Attributable)
- **Statement 2:** "...is used to build complex systems for applications like virtual assistants, robotics, and autonomous vehicles."
  - No nodes in the retrieved context support this statement.
  - **Verdict:** No (Not Attributable)

Calculate Contextual Recall:

- **Number of Attributable Statements:** 1 (Statement 1)
- **Total Number of Statements:** 2
- **Contextual Recall:** 1/2 = 0.5

Source: Dipanjan (DJ)

Contextual Relevancy

$$\text{Contextual Relevancy} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

Explanation of Key Variables:

- **Number of Relevant Statements:**
  - The count of statements in the retrieval context that are relevant to the input query.
- **Total Number of Statements:**
  - The total count of statements in the retrieval context.

How it Works:

- **Relevant Statements**:
  - The system, using an LLM as a judge, identifies statements in the retrieval context that are directly relevant to answering the input query.
- **Input Query**:
  - The specific question or topic the retrieval context aims to address.

- **Measures the relevancy of the information in the retrieved context document chunks (nodes) to the given input query**

- **Higher Context Relevance score represents a better retrieval system which can retrieve more semantically relevant nodes for queries**

Source: Dipanjan (DJ)

Contextual Relevancy

Example:

1. Input Query:
   - "What is AI?"
2. Retrieved Context:
   - **Node 1:** "NVIDIA makes chips for AI."
   - **Node 2:** "Google and Microsoft are battling out the market share for AI Chatbots."
   - **Node 3:** "Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning. AI includes applications like virtual assistants, robotics, and autonomous vehicles. It's evolving rapidly with advancements in machine learning and deep learning."
   - **Node 4:** "NLP is a branch of AI that enables computers to understand, interpret, and generate human language. Techniques include tokenization, stemming, and sentiment analysis. Applications range from chatbots to language translation services."
   - **Node 5:** "Machine learning is a field of artificial intelligence focused on enabling systems to learn patterns from data. Algorithms analyze past data to make predictions or classify information. Popular applications include recommendation systems and image recognition."

Metric Computation:

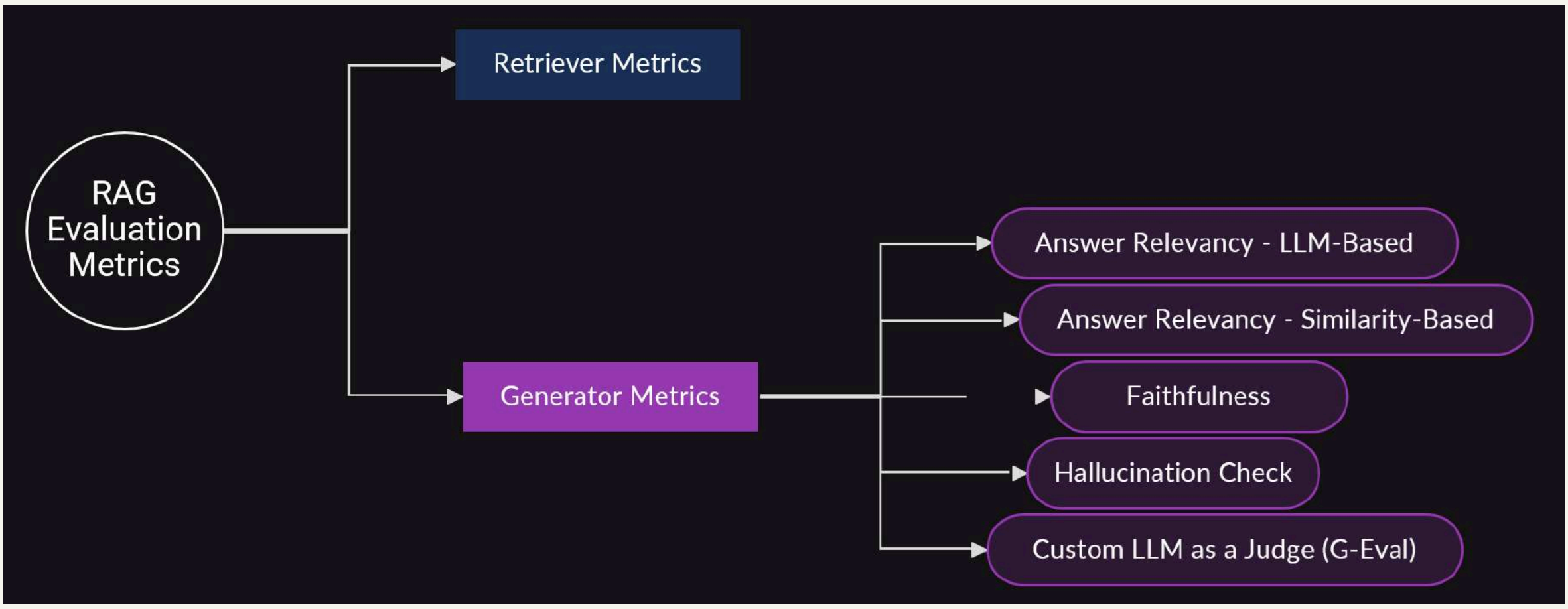
- Node 1:
  - Statements: ["NVIDIA makes chips for AI."]
  - Verdicts: [Not Relevant]
- Node 2:
  - Statements: ["Google and Microsoft are battling out the market share for AI Chatbots."]
  - Verdicts: [Not Relevant]
- Node 3:
  - Statements: ["Artificial intelligence refers to machines mimicking human intelligence, like problem solving and learning.", "AI includes applications like virtual assistants, robotics, and autonomous vehicles.", "It's evolving rapidly with advancements in machine learning and deep learning."]
  - Verdicts: [Relevant, Relevant, Relevant]
- Node 4:
  - Statements: ["NLP is a branch of AI that enables computers to understand, interpret, and generate human language.", "Techniques include tokenization, stemming, and sentiment analysis.", "Applications range from chatbots to language translation services."]
  - Verdicts: [Relevant, Relevant, Relevant]
- Node 5:
  - Statements: ["Machine learning is a field of artificial intelligence focused on enabling systems to learn patterns from data.", "Algorithms analyze past data to make predictions or classify information.", "Popular applications include recommendation systems and image recognition."]
  - Verdicts: [Relevant, Relevant, Relevant]

$$\text{Contextual Relevancy} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

Contextual Relevancy = 9/11 =0.8182

Source: Dipanjan (DJ)

Generator Metrics



- **Answer Relevancy (LLM Based):** Measures the relevancy of the information in the **generated response** to the provided **input query** using LLM as a Judge
- **Answer Relevancy (Similarity Based):** Measures the relevancy of the information in the **generated response** to the provided **input query** using semantic similarity between LLM generated queries from the response and the input query
- **Faithfulness:** Measures if the information in the **generated response** factually aligns with the contents of the **retrieved context** document chunks (nodes)
- **Hallucination Check:** Measures the proportion of contradictory statements by comparing the **generated response** to the **expected context** document chunks (ground truth reference)
- **Custom LLM-as-a-Judge:** G-Eval is a framework that uses **LLMs with chain-of-thoughts (CoT)** to evaluate LLM responses and retrieved contexts based on **ANY custom criteria**

Source: Dipanjan (DJ)

Answer Relevancy - LLM Based

$$\text{Answer Relevancy} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

Explanation of Key Variables:

- **Number of Relevant Statements:**
  - The count of statements in the actual output (generated response) that are relevant to the input query.
- **Total Number of Statements:**
  - The total count of statements in the actual output (generated response).

How it Works:

- **Relevant Statements**:
  - The evaluation system, using an LLM, identifies statements in the generated response that are directly relevant to answering the input query.
- **Input Query**:
  - The specific question or topic the generated response aims to address.

- **Measures the relevancy of the information in the generated response to the provided input query using LLM as a Judge**

- **Higher Answer Relevance shows the LLM Generator is able to generate better quality relevant responses for queries**

Source: Dipanjan (DJ)

Answer Relevancy - LLM Based

Example:

- Input Variables
  - Input Query: "What is AI?"
- **Generated Response:**
  - Statements: ["AI refers to machines mimicking human intelligence, such as problem-solving and learning.", "AI includes applications like virtual assistants, robotics, and autonomous vehicles."]

Metric Computation:

Computation:

Using the **Answer Relevancy** formula:

$$\text{Answer Relevancy} = \frac{\text{Number of Relevant Statements}}{\text{Total Number of Statements}}$$

1) Identify Relevant Statements:

- **Statement 1:** "AI refers to machines mimicking human intelligence, such as problem-solving and learning."
  - **Verdict:** Relevant
- **Statement 2:** "AI includes applications like virtual assistants, robotics, and autonomous vehicles."
  - **Verdict:** Relevant

2) Count Relevant Statements:

- Number of Relevant Statements: 2
- Total Number of Statements: 2

3) Calculate Answer Relevancy:

- **Answer Relevancy** = 2/2 = 1.0

Source: Dipanjan (DJ)

Answer Relevancy - Similarity Based

$$\text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N}\cos(E_{gi}, E_o) \quad \text{or equivalently,} \quad \text{Answer Relevancy} = \frac{1}{N}\sum_{i=1}^{N}\frac{E_{gi}\cdot E_o}{\|E_{gi}\|\|E_o\|}$$

Explanation of Key Variables:

- E_{gi}: Embedding of the ith generated question (reverse-engineered based on the response).
- E_o: Embedding of the original question (input query).
- N: Number of generated questions (typically defaults to 3).

How It Works:

- **Generate Variants**: Use an LLM to generate variants of the original question based on the answer (reverse engineering).
- **Calculate Cosine Similarity**: Compute the cosine similarity between each generated question's embedding and the original question's embedding.
- **Average Similarity**: Take the mean of these cosine similarity scores as the final **Answer Relevancy** score.

- **Measures the relevancy of the information in the generated response to the provided input query using semantic similarity between LLM generated queries from the response and the input query**

- **Higher Answer Relevance shows the LLM Generator is able to generate better quality relevant responses for queries**

Source: Dipanjan (DJ)

Answer Relevancy – Similarity Based

Example:

- Input Variables
  - **Input Query:** "What is AI?"
- **Generated Response:**
  - "AI refers to machines mimicking human intelligence, such as problem-solving and learning, and includes applications like virtual assistants, robotics, and autonomous vehicles."

Metric Computation:

1) **Generate Question Variants** (reverse-engineered from the generated response):

- Question 1: "What is the meaning of AI in terms of human-like intelligence?"
- Question 2: "What applications are included under AI technology?"
- Question 3: "How does AI mimic human intelligence?"

Using the **Answer Relevancy - Similarity-Based** formula:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{i=1}^{N} \cos(E_{g_i}, E_o)$$

2) **Calculate Cosine Similarity** between each generated question's embedding and the original question's embedding:

- Cosine similarity for Question 1 and Input Query = 0.92
- Cosine similarity for Question 2 and Input Query = 0.91
- Cosine similarity for Question 3 and Input Query = 0.93

3) **Average Similarity Score**:

- Answer Relevancy = (0.92 + 0.91 + 0.93)/3 = 0.9207

Source: Dipanjan (DJ)

Faithfulness

$$\text{Faithfulness} = \frac{\text{Number of Truthful Claims}}{\text{Total Number of Claims}}$$

Explanation of Key Variables:

- **Number of Truthful Claims:**
  - The count of claims in the actual output (generated response) that are verified as truthful based on the retrieval context.
- **Total Number of Claims:**
  - The total count of claims made in the actual output (generated response).

How it Works:

- **Identify Claims:**
  - The evaluation system, using an LLM, extracts all claims from the generated response.
- **Verify Truthfulness:**
  - Each claim is checked against the retrieval context to determine if it is consistent with the facts provided.
- **Calculate Faithfulness Score:**
  - The ratio of truthful claims to the total number of claims provides the **Faithfulness** score.

- **Measures if the information in the generated response factually aligns with the contents of the retrieved context document chunks (nodes)**

- **Higher Faithfulness means the generated response is more grounded with regard to the retrieved context reducing contradictions**

Source: Dipanjan (DJ)

Faithfulness

Example:

- Input Variables
  - **Input Query**: "What is AI?"
- Generated Response (Claims):
  - ["AI refers to machines mimicking human intelligence, including problem-solving and learning.", "AI has applications like virtual assistants, robotics, and autonomous vehicles."]
- Retrieved Context:
  - ["Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning. AI includes applications like virtual assistants, robotics, and autonomous vehicles. It's evolving rapidly with advancements in machine learning and deep learning.", "NLP is a branch of AI that enables computers to understand, interpret, and generate human language. Techniques include tokenization, stemming, and sentiment analysis. Applications range from chatbots to language translation services.", "Machine learning is a field of artificial intelligence focused on enabling systems to learn patterns from data. Algorithms analyze past data to make predictions or classify information. Popular applications include recommendation systems and image recognition."]

Metric Computation:

$$\text{Faithfulness} = \frac{\text{Number of Truthful Claims}}{\text{Total Number of Claims}}$$

1) Identify Claims in the Generated Response:

- Claim 1: "AI refers to machines mimicking human intelligence, including problem-solving and learning."
- Claim 2: "AI has applications like virtual assistants, robotics, and autonomous vehicles."

2) Verify Truthfulness by Checking Against Retrieved Context:

- Claim 1: Verified as truthful (matches context: "Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning.")
- Claim 2: Verified as truthful (matches context: "AI includes applications like virtual assistants, robotics, and autonomous vehicles.")

3) Count Truthful Claims:

- Number of Truthful Claims: 2
- Total Number of Claims: 2

4) Calculate Faithfulness Score:

- Faithfulness = 2/2 = 1.0

Source: Dipanjan (DJ)

Hallucination Check

$$\text{Hallucination} = \frac{\text{Number of Contradicted Contexts}}{\text{Total Number of Contexts}}$$

Explanation of Key Variables:

- **Number of Contradicted Contexts:**
  - The count of contexts in the ground truth where the actual output (generated response) presents information that directly contradicts the context.
- **Total Number of Contexts:**
  - The total count of contexts provided as ground truth for comparison.

How It Works:

- **Identify Contradictions:**
  - The evaluation system, using an LLM, examines each context to determine if any part of the generated response contradicts the information provided.
- **Calculate Hallucination Score:**
  - The ratio of contradicted contexts to the total number of contexts provides the **Hallucination** score.

- **Measures the proportion of contradictory statements by comparing the generated response to the expected context document chunks (ground truth reference)**

- **Lower the hallucination score, lower the proportion of contradictory statements making the response more grounded and relevant**

Source: Dipanjan (DJ)

Hallucination Check

Example:

- Input Variables
  - **Input Query**: "What is AI?"
- Generated Response:
  - "AI refers to machines mimicking human intelligence, such as problem-solving and learning, and includes applications like virtual assistants, robotics, and autonomous vehicles."
- Retrieved Context (Ground Truth):
  - ["Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning. AI includes applications like virtual assistants, robotics, and autonomous vehicles. It's evolving rapidly with advancements in machine learning and deep learning.", "Machine learning is a field of artificial intelligence focused on enabling systems to learn patterns from data. Algorithms analyze past data to make predictions or classify information. Popular applications include recommendation systems and image recognition."]

Metric Computation:

Using the Hallucination formula:

$$\text{Hallucination} = \frac{\text{Number of Contradicted Contexts}}{\text{Total Number of Contexts}}$$

1) Check for Contradictions in Each Context:

- **Context 1:** "Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning. AI includes applications like virtual assistants, robotics, and autonomous vehicles."
  - **Verdict**: No contradiction (The generated response aligns with this context)
- **Context 2:** "Machine learning is a field of artificial intelligence focused on enabling systems to learn patterns from data. Algorithms analyze past data to make predictions or classify information."
  - **Verdict**: No contradiction (The generated response does not contradict this context and focuses on AI in general, which is consistent)

2) Count Contradicted Contexts:

- Number of Contradicted Contexts: 0
- Total Number of Contexts: 2

3) Calculate Hallucination Score:

- Hallucination = 0/2 = 0.0

Source: Dipanjan (DJ)

Custom LLM as a Judge

How Is It Calculated?

G-Eval is a two-step algorithm that first generates a series of `evaluation_steps` using chain of thoughts (CoTs) based on the given `criteria`, before using the generated steps to determine the final score using the parameters presented in an `LLMTestCase`.

When you provide `evaluation_steps`, the `GEval` metric skips the first step and uses the provided steps to determine the final score instead.

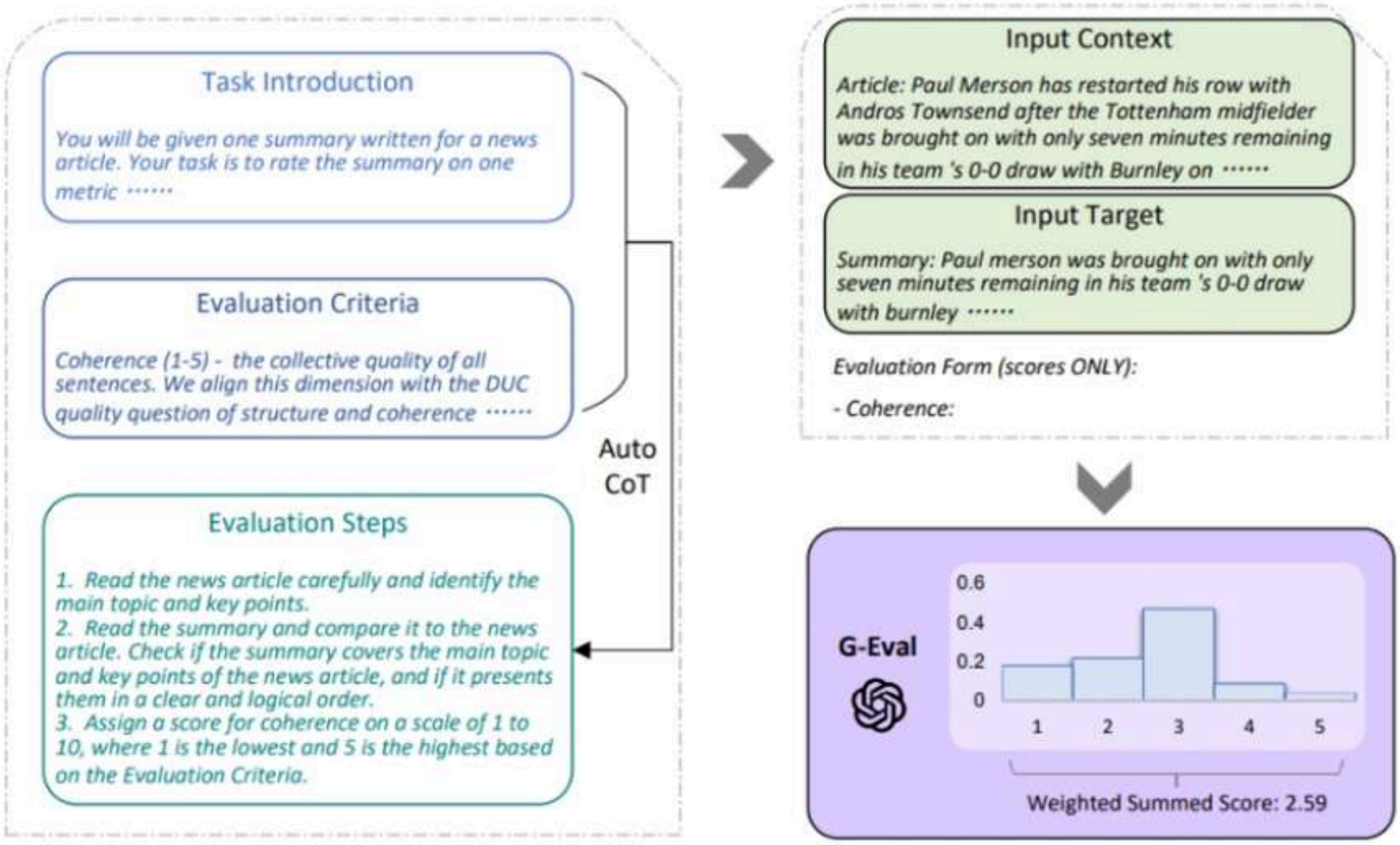


Figure 1: The overall framework of G-EVAL. We first input Task Introduction and Evaluation Criteria to the LLM, and ask it to generate a CoT of detailed Evaluation Steps. Then we use the prompt along with the generated CoT to evaluate the NLG outputs in a form-filling paradigm. Finally, we use the probability-weighted summation of the output scores as the final score.

- **G-Eval is a framework that uses LLMs with chain-of-thoughts (CoT) to evaluate LLM responses and retrieved contexts based on ANY custom criteria**

- **You can decide the judging criteria and give detailed evaluation steps using prompt instructions**

Source: Dipanjan (DJ)

Custom LLM as a Judge

```python
test_case = LLMTestCase(
    input='What is AI?',
    actual_output='AI refers to machines mimicking human intelligence, such as problem-solving and learning, and includes applications like electric sheep and cyborg kittens',
    retrieval_context=["Artificial intelligence refers to machines mimicking human intelligence, like problem-solving and learning. AI includes applications like virtual assistants, robotics, and autonomous vehicles. It's evolving rapidly with advancements in machine learning and deep learning.",.....]
)

metric = GEval(
    threshold=0.5,
    model="gpt-4o",
    name="RAG Fact Checker",
    evaluation_steps=[
        "Create a list of statements from 'actual output'",
        "Validate if they are relevant and answers the given question in 'input', penalize if any statements are irrelevant",
        "Also Validate if they exist in 'expected output', penalize if any statements are missing or factually wrong",
        "Also validate if these statements are grounded in the 'retrieval context' and penalize if they are missing or factually wrong",
        "Finally also penalize if any statements seem to be invented or made up and do not make sense factually given the 'input' and 'retrieval context'"
    ],
    evaluation_params=[LLMTestCaseParams.INPUT,
                       LLMTestCaseParams.ACTUAL_OUTPUT,
                       LLMTestCaseParams.RETRIEVAL_CONTEXT],
    verbose_mode=True
)

result = evaluate([test_case], [metric])
```

```
RAG Fact Checker (GEval) Verbose Logs
**************************************************

Criteria:
None

Evaluation Steps:
[
    "Create a list of statements from 'actual output'",
    "Validate if they are relevant and answers the given question in 'input', penalize if any statements are irrelevant",
    "Also Validate if they exist in 'expected output', penalize if any statements are missing or factually wrong",
    "Also validate if these statements are grounded in the 'retrieval context' and penalize if they are missing or factually wrong",
    "Finally also penalize if any statements seem to be invented or made up and do not make sense factually given the 'input' and 'retrieval context'"
]

Score: 0.5326435311680616
Reason: The actual output partially aligns with the input and retrieval context. It correctly defines AI as machines mimicking human intelligence, which is consistent with the retrieval context. However, it includes irrelevant and potentially made-up examples like 'electric sheep and cyborg kittens' not supported by the context or expected output.
```

Source: Dipanjan (DJ)

RAG Evaluation Example with DeepEval

```
from deepeval import evaluate
from deepeval.metrics import ContextualPrecisionMetric, ContextualRecallMetric, ContextualRelevancyMetric
from deepeval.metrics import AnswerRelevancyMetric, FaithfulnessMetric, HallucinationMetric
from deepeval.metrics.ragas import RAGASAnswerRelevancyMetric

eval_dataset.test_cases = [....] # create your test cases
contextual_precision = ContextualPrecisionMetric(threshold=0.5, include_reason=True, model="gpt-4o")
contextual_recall = ContextualRecallMetric(threshold=0.5, include_reason=True, model="gpt-4o")
contextual_relevancy = ContextualRelevancyMetric(threshold=0.5, include_reason=True, model="gpt-4o")
answer_relevancy = AnswerRelevancyMetric(threshold=0.5, include_reason=True, model="gpt-4o")
faithfulness = FaithfulnessMetric(threshold=0.5, include_reason=True, model="gpt-4o")
hallucination = HallucinationMetric(threshold=0.5, include_reason=True, model="gpt-4o")
ragas_answer_relevancy = RAGASAnswerRelevancyMetric(threshold=0.5, embeddings=OpenAIEmbeddings(),
                                                    model="gpt-4o")

eval_results = evaluate(test_cases=eval_dataset.test_cases,
                        metrics=[contextual_precision, contextual_recall, contextual_relevancy,
                                 answer_relevancy, ragas_answer_relevancy, faithfulness, hallucination])


## EVAL OUTPUT ##

Evaluating 10 test case(s) in parallel: |██████████|100% (10/10) [Time Taken: 00:39,  3.98s/test case]
======================================================================

Metrics Summary

  - ✅ Contextual Precision (score: 1.0, threshold: 0.5, strict: False, ....)
  - ❌ Contextual Recall (score: 0.25, threshold: 0.5, strict: False, ....)
  - ❌ Contextual Relevancy (score: 0.3333333333333333, threshold: 0.5, strict: False, ....)
  - ✅ Answer Relevancy (score: 1.0, threshold: 0.5, strict: False, ....)
  - ❌ Answer Relevancy (ragas) (score: 0.0, threshold: 0.5, strict: False, ....)
  - ✅ Faithfulness (score: 1.0, threshold: 0.5, strict: False, ....)
  - ❌ Hallucination (score: 1.0, threshold: 0.5, strict: False, ....)
```

- **You can leverage libraries like DeepEval and Ragas to make things easier for you or even create your own custom eval metrics**

Source: Dipanjan (DJ)