1. Langchain Evaluation
2. Langsmith
3. Ragas
4. Truera – triad metrics
5. Bedrock Evaluation
6. RAG-checker
7. Deepeval
8. Llama-index evaluation
9. Evaluating **Multi-Modal RAG**

# RAGAS

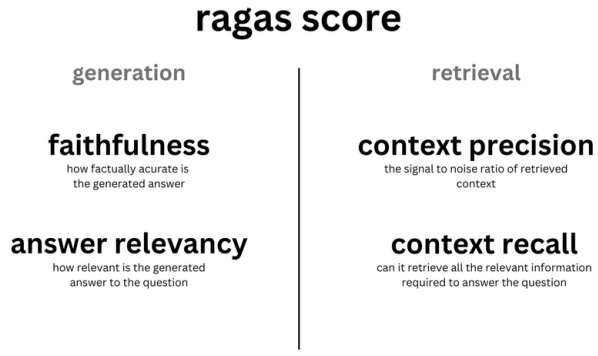
ragas is a framework that helps you evaluate your Retrieval Augmented Generation (RAG) pipelines.. ragas can be integrated with your CI/CD to provide continuous checks to ensure performance.

Ragas provides several metrics to evaluate various aspects of your RAG systems:

1. Retriever: Offers `context_precision` and `context_recall` that measure the performance of your retrieval system.
2. Generator (LLM): Provides `faithfulness` that measures hallucinations and `answer_relevancy` that measures how relevant the answers are to the question.

Four metrics:

- **Faithfulness** - Measures the factual consistency of the answer to the context based on the question.
- **Context_precision** - Measures how relevant the retrieved context is to the question, conveying the quality of the retrieval pipeline.
- **Answer_relevancy** - Measures how relevant the answer is to the question.
- **Context_recall** - Measures the retriever's ability to retrieve all necessary information required to answer the question.



## :fire: Quickstart

```
from ragas import evaluate

from datasets import Dataset

import os

os.environ["OPENAI_API_KEY"] = "your-openai-key"

# prepare your huggingface dataset in the format

# Dataset({

#     features: ['question', 'contexts', 'answer'],

#     num_rows: 25

# })


dataset: Dataset

results = evaluate(dataset)

# {'ragas_score': 0.860, 'context_relevancy': 0.817,

# 'faithfulness': 0.892, 'answer_relevancy': 0.874}
```

Ragas measures your pipeline's performance against different dimensions

1. **Faithfulness**: This measures the factual consistency of the generated answer against the given context. It is calculated from answer and retrieved context. The answer is scaled to (0,1) range. Higher the better.

The generated answer is regarded as faithful if all the claims made in the answer can be inferred from the given context. To calculate this, a set of claims from the generated answer is first identified. Then each of these claims is cross-checked with the given context to determine if it can be inferred from the context

$$\text{Faithfulness score} = \frac{|\text{Number of claims in the generated answer that can be inferred from given context}|}{|\text{Total number of claims in the generated answer}|}$$

Let's examine how faithfulness was calculated using the low faithfulness answer:

- **Step 1:** Break the generated answer into individual statements.
    - Statements:

    - Statement 1: “Einstein was born in Germany.”
    - Statement 2: “Einstein was born on 20th March 1879.”
- **Step 2:** For each of the generated statements, verify if it can be inferred from the given context.
    - Statement 1: Yes
    - Statement 2: No
- **Step 3:** Use the formula depicted above to calculate faithfulness.

Faithfullness = ½=0.5.

2. **Context Precision/Relevancy**: measures how relevant retrieved contexts are to the question. Ideally, the context should only contain information necessary to answer the question. The presence of redundant information in the context is penalized.

Context Precision is a metric that evaluates whether all of the ground-truth relevant items present in the **`contexts`** are ranked higher or not. Ideally all the relevant chunks must appear at the top ranks. This metric is computed using the **`question`**, **`ground_truth`** and the **`contexts`**, with values ranging between 0 and 1, where higher scores indicate better precision.

$$\text{Context Precision@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$

Where K is the total number of chunks in `contexts` and vk∈{0,1} is the relevance indicator at rank k.

Let’s examine how context precision was calculated using the low context precision example:

**Step 1**: For each chunk in retrieved context, check if it is relevant or not relevant to arrive at the ground truth for the given question.

**Step 2**: Calculate precision@k for each chunk in the context.

$$\text{Precision@1} = \frac{0}{1} = 0$$

$$\text{Precision@2} = \frac{1}{2} = 0.5$$

**Step 3**: Calculate the mean of precision@k to arrive at the final context precision score.

$$\text{Context Precision} = \frac{(0+0.5)}{1} = 0.5$$

3. **Context Recall****: measures the recall of the retrieved context using annotated answer as ground truth. Annotated answer is taken as proxy for ground truth context.

Context recall measures the extent to which the retrieved context aligns with the annotated answer, treated as the ground truth. It is computed using **`question`**, **`ground truth`** and the retrieved **`context`**, and the values range between 0 and 1, with higher values indicating better performance. To estimate context recall from the ground truth answer, each claim in the ground truth answer is analyzed to determine whether it can be attributed to the retrieved context or not.

In an ideal scenario, all claims in the ground truth answer should be attributable to the retrieved context. A reference free version of this is available as **context_utilization**.

$$\text{context recall} = \frac{|\text{GT claims that can be attributed to context}|}{|\text{Number of claims in GT}|}$$

Let's examine how context recall was calculated using the low context recall example:

- **Step 1:** Break the ground truth answer into individual statements.
  - Statements:
    - Statement 1: "France is in Western Europe."
    - Statement 2: "Its capital is Paris."
- **Step 2:** For each of the ground truth statements, verify if it can be attributed to the retrieved context.
  - Statement 1: Yes
  - Statement 2: No
- **Step 3:** Use the formula depicted above to calculate context recall

$$\text{context recall} = \frac{1}{2} = 0.5$$

4. **Answer Relevancy**: refers to the degree to which a response directly addresses and is appropriate for a given question or context.

This **does not take the factuality of the answer** into consideration but rather **penalizes the present of redundant information or incomplete answers** given a question.

The evaluation metric, Answer Relevancy, focuses on assessing how pertinent the generated answer is to the given prompt. A lower score is assigned to answers that are incomplete or contain redundant information and higher scores indicate better relevancy. This metric is computed using the `question`, the `context` and the `answer`.

The Answer Relevancy is defined as the mean cosine similarity of the original question to a number of artifical questions, which where generated (reverse engineered) based on the answer:

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

$$\text{answer relevancy} = \frac{1}{N}\sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\|E_{g_i}\|\|E_o\|}$$

Where:

- Egi is the embedding of the generated question i.
- Eo is the embedding of the original question.
- N is the number of generated questions, which is 3 default.

Please note, that even though in practice the score will range between 0 and 1 most of the time, this is not mathematically guaranteed, due to the nature of the cosine similarity ranging from -1 to 1.

**Note** : This is reference free metric. If you're looking to compare ground truth answer with generated answer refer to answer_correctness

An answer is deemed relevant when it directly and appropriately addresses the original question. Importantly, our assessment of answer relevance does not consider factuality but instead penalizes cases where the answer lacks completeness or contains redundant details.

To calculate this score, the **LLM is prompted to generate an appropriate question for the generated answer multiple times, and the mean cosine similarity between these generated questions and the original question is measured**.

The underlying idea is that if the generated answer accurately addresses the initial question, the LLM should be able to generate questions from the answer that align with the original question.

To calculate the relevance of the answer to the given question, we follow two steps:

- **Step 1:** Reverse-engineer 'n' variants of the question from the generated answer using a Large Language Model (LLM). For instance, for the first answer, the LLM might generate the following possible questions:
  - *Question 1:* "In which part of Europe is France located?"
  - *Question 2:* "What is the geographical location of France within Europe?"
  - *Question 3:* "Can you identify the region of Europe where France is situated?"
- **Step 2:** Calculate the mean cosine similarity between the generated questions and the actual question.

The underlying concept is that if the answer correctly addresses the question, it is highly probable that the original question can be reconstructed solely from the answer.

5. **Context utilization** is like a reference free version of context_precision metrics.

Context utilization is a metric that evaluates whether all of the answer relevant items present in the `contexts` are ranked higher or not. Ideally all the relevant chunks must appear at the top ranks. This metric is computed using the `question`, `answer` and the `contexts`, with values ranging between 0 and 1, where higher scores indicate better precision.

$$\text{Context Utilization@K} = \frac{\sum_{k=1}^{K} (\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{(\text{true positives@k} + \text{false positives@k})}$$

Where K is the total number of chunks in `contexts` and vk∈{0,1} is the relevance indicator at rank k.

Let's examine how context utilization was calculated using the low context utilization example:

**Step 1**: For each chunk in retrieved context, check if it is relevant or not relevant to arrive at the answer for the given question.

**Step 2**: Calculate precision@k for each chunk in the context.

Precision@1=0/1=0
Precision@2=1/2=0.5

**Step 3**: Calculate the mean of precision@k to arrive at the final context utilization score.

$$\text{Context Utilization} = \frac{(0+0.5)}{1} = 0.5$$

6. **Context Entities Recall:** It measure recall of the retrieved context, based on the number of entities present in both ground_truths and contexts relative to the number of entities present in the `ground_truths` alone.

Simply put, it is a measure of what fraction of entities are recalled from `ground_truths`. This metric is useful in fact-based use cases like tourism help desk, historical QA, etc. This metric can help evaluate the retrieval mechanism for entities, based on comparison with entities present in `ground_truths`, because in cases where entities matter, we need the `contexts` which cover them.

To compute this metric, we use two sets, GE and CE, as set of entities present in `ground_truths` and set of entities present in `contexts` respectively. We then take the number of elements in intersection of these sets and divide it by the number of elements present in the GE, given by the formula:

$$\text{context entity recall} = \frac{|CE \cap GE|}{|GE|}$$

Let us consider the ground truth and the contexts given above.

- **Step-1**: Find entities present in the ground truths.

Entities in ground truth (GE) - ['Taj Mahal', 'Yamuna', 'Agra', '1631', 'Shah Jahan', Mumtaz Mahal']

- **Step-2**: Find entities present in the context.

Entities in context (CE1) - ['Taj Mahal', 'Agra', 'Shah Jahan', 'Mumtaz Mahal', 'India']

Entities in context (CE2) - ['Taj Mahal', 'UNESCO', 'India']

- **Step-3**: Use the formula given above to calculate entity-recall

$$\text{context entity recall - 1} = \frac{|CE1 \cap GE|}{|GE|} = 4/6 = 0.666$$

$$\text{context entity recall - 2} = \frac{|CE2 \cap GE|}{|GE|} = 1/6 = 0.166$$

We can see that the first context had a high entity recall, because it has a better entity coverage given the ground truth. If these two contexts were fetched by two retrieval mechanisms on same set of documents, we could say that the first mechanism was better than the other in use-cases where entities are of importance.

7. **Noise Sensitivity:** Noise sensitivity measures how often a system makes errors by providing incorrect responses when utilizing either relevant or irrelevant retrieved documents.

The score ranges from 0 to 1, with lower values indicating better performance. Noise sensitivity is computed using the question, ground truth, answer, and the retrieved context.

To estimate noise sensitivity, each claim in the generated answer is examined to determine whether it is correct based on the ground truth and whether it can be attributed to the relevant (or irrelevant) retrieved context. Ideally, all claims in the answer should be supported by the relevant retrieved context

$$\text{noise sensitivity (relevant)} = \frac{|\text{Number of incorrect claims in answer}|}{|\text{Number of claims in the Answer}|}$$

Let's examine how noise sensitivity in relevant context was calculated:

- **Step 1:** Identify the relevant contexts from which the ground truth can be inferred.

  Ground Truth: The Life Insurance Corporation of India (LIC) is the largest insurance company in India, established in 1956 through the nationalization of the insurance industry. It is known for managing a large portfolio of investments.

  - **Contexts**:

- Context 1 : `The Life Insurance Corporation of India (LIC) was established in 1956` following the nationalization of the insurance industry in India.
- Context 2: `LIC is the largest insurance company in India`, with a vast network of policyholders and a significant role in the financial sector.
- Context 3: `As the largest institutional investor in India, LIC manages a substantial funds`, contributing to the financial stability of the country.

- **Step 2:** Verify if the claims in the generated answer can be inferred from the relevant context.

  Answer: The Life Insurance Corporation of India (LIC) is the largest insurance company in India, known for its vast portfolio of investments. LIC contributs to the financial stability of the country.

  - **Contexts**:

- Context 1: The Life Insurance Corporation of India (LIC) was established in 1956 following the nationalization of the insurance industry in India.
- Context 2: `LIC is the largest insurance company in India`, with a vast network of policyholders and a significant role in the financial sector.
- Context 3: `As the largest institutional investor in India, LIC manages a substantial funds, contributing to the financial stability of the country.`

- **Step 3:** Identify any incorrect claims in the answer (i.e., answer statements that are not supported by the ground truth).
  - Ground Truth: The Life Insurance Corporation of India (LIC) is the largest insurance company in India, established in 1956 through the nationalization of the insurance industry. It is known for managing a large portfolio of investments.
  - Answer: The Life Insurance Corporation of India (LIC) is the largest insurance company in India, known for its vast portfolio of investments. `LIC contributs to the financial stability of the country.`

  Explanation: The ground truth does not mention anything about LIC contributing to the financial stability of the country. Therefore, this statement in the answer is incorrect.

  Incorrect Statement: 1

  Total claims: 3

- **Step 4:** Calculate noise sensitivity using the formula:

noise sensitivity=1/3=0.333

This results in a noise sensitivity score of 0.333, indicating that one out of three claims in the answer was incorrect.

8. **Answer semantic similarity:** pertains to the assessment of the semantic resemblance between the generated answer and the ground truth.

This evaluation is based on the **`ground truth`** and the **`answer`**, with values falling within the range of 0 to 1. A higher score signifies a better alignment between the generated answer and the ground truth.

Measuring the semantic similarity between answers can offer valuable insights into the quality of the generated response. This evaluation utilizes a cross-encoder model to calculate the semantic similarity score.

Let's examine how answer similarity was calculated for the first answer:

- **Step 1:** Vectorize the ground truth answer using the specified embedding model.
- **Step 2:** Vectorize the generated answer using the same embedding model.
- **Step 3:** Compute the cosine similarity between the two vectors.

9. **Answer Correctness:** Involves gauging the accuracy of the generated answer when compared to the ground truth.

This evaluation relies on the `ground truth` and the `answer`, with scores ranging from 0 to 1. A higher score indicates a closer alignment between the generated answer and the ground truth, signifying better correctness.

Answer correctness encompasses two critical aspects: **semantic similarity** between the generated answer and the ground truth, as well as **factual similarity**. These aspects are combined using a weighted scheme to formulate the answer correctness score. Users also have the option to employ a 'threshold' value to round the resulting score to binary, if desired.

Let's calculate the answer correctness for the answer with low answer correctness. It is computed as the sum of factual correctness and the semantic similarity between the given answer and the ground truth.

Factual correctness quantifies the factual overlap between the generated answer and the ground truth answer. This is done using the concepts of:

- **TP** (True Positive): Facts or statements that are present in both the ground truth and the generated answer.
- **FP** (False Positive): Facts or statements that are present in the generated answer but not in the ground truth.
- **FN** (False Negative): Facts or statements that are present in the ground truth but not in the generated answer.

In the second example:

- TP: `[Einstein was born in 1879]`
- FP: `[Einstein was born in Spain]`
- FN: `[Einstein was born in Germany]`

Now, we can use the formula for the F1 score to quantify correctness based on the number of statements in each of these lists:

$$\text{F1 Score} = \frac{|\text{TP}}{(|\text{TP}| + 0.5 \times (|\text{FP}| + |\text{FN}|))}$$

Next, we calculate the semantic similarity between the generated answer and the ground truth. Read more about it here.

Once we have the semantic similarity, we take a weighted average of the semantic similarity and the factual similarity calculated above to arrive at the final score. You can adjust this weightage by modifying the `weights` parameter.

10. **Aspect Critiques:** Designed to judge the submission against defined aspects like harmlessness, correctness, etc. You can also define your own aspect and validate the submission against your desired aspect. The output of aspect critiques is always binary.

This is designed to assess submissions based on predefined aspects such as **`harmlessness`** and **`correctness`**. Additionally, users have the flexibility to define their own aspects for evaluating submissions according to their specific criteria. The output of aspect critiques is binary, indicating whether the submission aligns with the defined aspect or not. This evaluation is performed using the 'answer' as input.

Critiques within the LLM evaluators evaluate submissions based on the provided aspect. Ragas Critiques offers a range of predefined aspects like correctness, harmfulness, etc.

(Please refer to `SUPPORTED_ASPECTS` for a complete list). If you prefer, you can also create custom aspects to evaluate submissions according to your unique requirements.

The `strictness` parameter plays a crucial role in maintaining a certain level of self-consistency in predictions, with an ideal range typically falling between 2 to 4

The final `ragas_score` is the harmonic mean of individual metric scores.

1. Why harmonic mean?

Harmonic-Mean penalizes extreme values. For example, if your generated answer is fully factually consistent with the context (faithfulness = 1) but is not relevant to the question (relevancy = 0), a simple average would give you a score of 0.5 but a harmonic mean will give you 0.0

2. How to use Ragas to improve your pipeline?

Here we assume that you already have your RAG pipeline ready. When it comes to RAG pipelines, there are mainly two parts - Retriever and generator. A change in any of these should also impact your pipelines' quality.

- First, decide on one parameter that you're interested in adjusting. for example the number of retrieved documents, K.
- Collect a set of sample prompts (min 20) to form your test set.
- Run your pipeline using the test set before and after the change. Each time record the prompts with context and generated output.
- Run ragas evaluation for each of them to generate evaluation scores.
- Compare the scores and you will know how much the change has affected your pipelines' performance.