Index

⚠️ Instruction before running this notebook

- Paste your huggingface api token in the .env file
- Install the requirements using `pip install -r requirements.txt`
- Run the command `python -m spacy download en`. This will download the default English language model for the SpaCy library for extracting the entities.

Introduction

For evaluating the responses of our ChatBot, I have established an evaluation mechanism centered around four key metrics:

- **Answer Similarity**
- **Faithfulness**
- **Relevance**
- **Answer Correctness**

How this works:

- We have created an LLM Evaluator for each metric, incorporating specific evaluation criteria and a scoring rubric ranging from **1 to 5** (where **1** denotes the lowest score and **5** denotes the highest score).
- This criteria and scoring rubric are provided to an Evaluator LLM along with the generated response and the retrieved context from which the response is generated. After processing the input, the Evaluator LLM outputs:
  1. A numeric score between **1 and 5**
  2. Feedback explaining why this particular score was given

Each metric and its scoring criteria are defined in detail later in the report.

Library Imports

In [ ]:
```python
import warnings, os
warnings.filterwarnings('ignore')

from tqdm.auto import tqdm
import warnings, os
import pandas as pd
import numpy as np
import time
from sklearn.model_selection import train_test_split
```

Preparing Question to be asked to ChatBot

Here I'm extracting few question from the data to be asked to the ChatBot

In [ ]:
```python
# Splitting the questions to be asked to chatbot
aws_faq = pd.read_csv('../aws_faqs.csv')
X_train, X_test = train_test_split(aws_faq, test_size=0.02, random_state=42)
aws_faq = X_test
```

Generate Answers using Chatbot

Methodology:

- Send each questions from the data frame to ChatBot.
- ChatBot will generate the answers and return the generated answer.
- Create a data frame named `chatbot_response_df` containing the generated answers and its relevant docs from the vector store.

In [ ]:
```python
# Importing the chatpot
from aws_faq_chatbot import AwsFaqChatBot
chatbot = AwsFaqChatBot()
```

```
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-git-credential` if using via `huggingface-cli` if you want
to set the git credential as well.
Token is valid (permission: write).
Your token has been saved to C:\Users\tapas\.cache\huggingface\token
Login successful
WARNING:tensorflow:From c:\Users\tapas\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Pleas
e use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [ ]:
```python
# Generating answers and store it in the dataframe
chatbot_output = []
for row in tqdm(aws_faq.iterrows()):
    start_time = time.time()
    answer, relevant_docs = chatbot.ask(row[1]['question'])
    end_time = time.time()

    chatbot_output.append({
        'question': row[1]['question'],
        'generated_answer': answer,
        'context': relevant_docs,
        'latency(seconds)': round((end_time - start_time), 2)})

chatbot_response_df = pd.DataFrame(chatbot_output)
```

```
13it [00:37,  2.92s/it]
```

ChatBot Response Evaluation

Following are the 5 metrics for evaluating the chatbor response:

- Latency
- Answer Similarity
- Fithfullness
- Relevance
- Answer Correctness

Latency

In [ ]:
```python
# Restrincting to only 5 output for the sake of readability
for row in chatbot_response_df.head().iterrows():
    print(f"Question: {row[1]['question']}")
    print(f"Answer: {row[1]['generated_answer']}")
    print(f"Context: {row[1]['context']}")
    print(f"Latency(seconds): {row[1]['latency(seconds)']}")
    print('==================================================\n')
```

```
Question: Q. Can I tag a Spot Fleet request?
Answer:  Yes, you can tag a Spot Fleet request when you create it. However, the Spot Fleet itself cannot be tagged.
Context: ['You can request to launch Spot Instances with tags via Spot Fleet. The Fleet by itself cannot be tagged.']
Latency(seconds): 3.18
==================================================

Question: Q: Where can I learn more about EFS?
Answer:
    You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.
Context: ['You can visit the Amazon EFS FAQ pageAmazon EFS FAQ page.']
Latency(seconds): 1.43
==================================================

Question: Q: Do RIs provide a capacity reservation?
Answer:  A: Yes, RIs do provide a capacity reservation. As per the context, to take advantage of the capacity reservation, one should buy a Reserved Instance (RI) in a specific Availability Z
one (AZ).
Context: ['If you want to take advantage of the capacity reservation, then you should buy an RI in a specific AZ.']
Latency(seconds): 3.89
==================================================

Question: Q: What happens to my data when a system terminates?
Answer:
    When a system terminates, the root device and attached device data are stored on the corresponding EBS volumes. However, memory (RAM) contents are not saved. Therefore, any data stored in
the memory will be lost.
Context: ['As with the Stop feature, root device and attached device data are stored on the corresponding EBS volumes. Memory (RAM) contents are stored on the EBS root volume.']
Latency(seconds): 2.35
==================================================

Question: Q: Can I still use/add more Previous Generation instances?
Answer:  Yes, you can still use and add more Previous Generation instances. They are not going away.
Context: ['No. Your Reserved Instances will not change, and the Previous Generation instances are not going away.']
Latency(seconds): 1.03
==================================================
```

In [ ]:
```python
# calculating the overall latency of the ChatBot
chatbot_response_df['latency(seconds)'] = chatbot_response_df['latency(seconds)'].astype(float)

print("Overall Latency of the ChatBot:"
f" {round(np.mean(chatbot_response_df['latency(seconds)']), 2)} Seconds")
```

```
Overall Latency of the ChatBot: 2.92 Seconds
```

In [ ]:
```python
# Saving generated response .csv file
chatbot_response_df.to_csv('./scores/chatbot_response.csv', index=False)
```

Answer Similarity

Answer Similarity answers the question: **Is the response semantically similar based on the context?**

Answer similarity is scored based on how closely the output's meaning matches the target, with higher scores for greater alignment.

Scoring Criteria

- *Score 1:* The response has little to no semantic similarity to the reference answer.
- *Score 2:* The response displays partial semantic similarity to the reference answer on some aspects.
- *Score 3:* The response has moderate semantic similarity to the reference answer.
- *Score 4:* The response aligns with the reference answer in most aspects and has substantial semantic similarity.
- *Score 5:* The response closely aligns with the reference answer in all significant aspects.

In [ ]:
```python
from metrics.answer_similarity import AnswerSimilarity

ans_similarity = AnswerSimilarity()
answer_similarity_scores = ans_similarity.evaluate(chatbot_response_df)
```

```
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-git-credential` if using via `huggingface-cli` if you want
to set the git credential as well.
Token is valid (permission: write).
Your token has been saved to C:\Users\tapas\.cache\huggingface\token
Login successful
13it [00:50,  3.86s/it]
```

Similarity Score result

In [ ]:
```python
# Restrincting to only 2 output for the sake of readability
for row in answer_similarity_scores.head(2).iterrows():
    print(f"Question: {row[1]['question']}")
    print(f"Answer: {row[1]['generated_answer']}")
    print(f"Context: {row[1]['context']}")
    print(f"Similarity Score: {row[1]['similarity_score']}")
    print(f"{row[1]['similarity_score_feedback']}")
    print('==================================================\n')
```

```
Question: Q. Can I tag a Spot Fleet request?
Answer:  Yes, you can tag a Spot Fleet request when you create it. However, the Spot Fleet itself cannot be tagged.
Context: ['You can request to launch Spot Instances with tags via Spot Fleet. The Fleet by itself cannot be tagged.']
Similarity Score: 5
Feedback: The response correctly identifies that Spot Fleet requests can be tagged during creation, and also clarifies that the Spot Fleet itself cannot be tagged. This is semantically simila
r to the reference answer, which states that Spot Instances can be launched with tags via Spot Fleet, and that the Fleet itself cannot be tagged.
==================================================

Question: Q: Where can I learn more about EFS?
Answer:
    You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.
Context: ['You can visit the Amazon EFS FAQ pageAmazon EFS FAQ page.']
Similarity Score: 5
Feedback: The provided response, 'You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.', has a high degree of semantic similarity to the reference answer, 'You can visit t
he Amazon EFS FAQ pageAmazon EFS FAQ page.'. Both provide the same information and direct the user to the Amazon EFS FAQ page.
==================================================
```

In [ ]:
```python
# calculating the overall Similarity score of the ChatBot
answer_similarity_scores['similarity_score'] = answer_similarity_scores['similarity_score'].astype(int)

print("Overall Similarity Score Rating of the ChatBot:"
f" {round(np.mean(answer_similarity_scores['similarity_score']), 2)}")
```

```
Overall Similarity Score Rating of the ChatBot: 4.08
```

In [ ]:
```python
# Saving to similarity scores .csv file
answer_similarity_scores.to_csv('./scores/answer_similarity_scores.csv', index=False)
```

Faithfullness

Faithfullness answers the question: **Does the response accurately reflect the given context?**

Faithfulness checks if the response is factually consistent with the context.

Scoring Criteria

- *Score 1:* None of the claims in the response can be inferred from the provided context.
- *Score 2:* Some of the claims in the response can be inferred from the provided context, but the majority of the response is missing from, inconsistent with, or contradictory to the provided context.
- *Score 3:* Half or more of the claims in the response can be inferred from the provided context.
- *Score 4:* Most of the claims in the response can be inferred from the provided context, with very little information that is not directly supported by the provided context.
- *Score 5:* All of the claims in the response are directly supported by the provided context, demonstrating high faithfulness to the provided context.

In [ ]:
```python
from metrics.faithfullness import Faithfullness

faithfullness = Faithfullness()
faithfullness_scores = faithfullness.evaluate(chatbot_response_df)
```

```
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-git-credential` if using via `huggingface-cli` if you want
to set the git credential as well.
Token is valid (permission: write).
Your token has been saved to C:\Users\tapas\.cache\huggingface\token
Login successful
13it [01:01,  4.75s/it]
```

Faithfullness result

In [ ]:
```python
# Restrincting to only 2 output for the sake of readability
for row in faithfullness_scores.head(2).iterrows():
    print(f"Question: {row[1]['question']}")
    print(f"Answer: {row[1]['generated_answer']}")
    print(f"Context: {row[1]['context']}")
    print(f"Faithfullness Score: {row[1]['faithfullness_score']}")
    print(f"{row[1]['faithfullness_score_feedback']}")
    print('==================================================\n')
```

```
Question: Q. Can I tag a Spot Fleet request?
Answer:  Yes, you can tag a Spot Fleet request when you create it. However, the Spot Fleet itself cannot be tagged.
Context: ['You can request to launch Spot Instances with tags via Spot Fleet. The Fleet by itself cannot be tagged.']
Faithfullness Score: 5
Feedback: The response correctly identifies that Spot Fleet requests can be tagged when created, and correctly identifies that the Spot Fleet itself cannot be tagged. These claims are directl
y supported by the context.
==================================================

Question: Q: Where can I learn more about EFS?
Answer:
    You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.
Context: ['You can visit the Amazon EFS FAQ pageAmazon EFS FAQ page.']
Faithfullness Score: 5
Feedback: The provided response accurately directs the user to learn more about Amazon EFS by visiting the Amazon EFS FAQ page, which is directly supported by the provided context.
==================================================
```

In [ ]:
```python
# calculating the overall Faithfullness Score of the ChatBot
faithfullness_scores['faithfullness_score'] = faithfullness_scores['faithfullness_score'].astype(int)

print("Overall Faithfullness Score Rating of the ChatBot:"
f" {round(np.mean(faithfullness_scores['faithfullness_score']), 2)}")
```

```
Overall Faithfullness Score Rating of the ChatBot: 4.31
```

In [ ]:
```python
# Saving to faithfullness scores .csv file
faithfullness_scores.to_csv('./scores/faithfullness_scores.csv', index=False)
```

Relevance

Relevance answers the question: **Does the response address the given question and context?**

Relevance measures how well the response addresses the question and context. Scores reflect how directly the output answers the input question within the context.

Scoring Criteria

- *Score 1:* The response doesn't mention anything about the question or is completely irrelevant to the provided context.
- *Score 2:* The response provides some relevance to the question and is somehow related to the provided context.
- *Score 3:* The response mostly answers the question and is largely consistent with the provided context.
- *Score 4:* The response answers the question and is consistent with the provided context.
- *Score 5:* The response answers the question comprehensively using the provided context.

In [ ]:
```python
from metrics.relevance import Relevance

relevance = Relevance()
relevance_scores = relevance.evaluate(chatbot_response_df)
```

```
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-git-credential` if using via `huggingface-cli` if you want
to set the git credential as well.
Token is valid (permission: write).
Your token has been saved to C:\Users\tapas\.cache\huggingface\token
Login successful
13it [01:12,  5.60s/it]
```

Relevance result

In [ ]:
```python
# Restrincting to only 2 output for the sake of readability
for row in relevance_scores.head(2).iterrows():
    print(f"Question: {row[1]['question']}")
    print(f"Answer: {row[1]['generated_answer']}")
    print(f"Context: {row[1]['context']}")
    print(f"Relevance Score: {row[1]['relevance_score']}")
    print(f"{row[1]['relevance_score_feedback']}")
    print('==================================================\n')
```

```
Question: Q. Can I tag a Spot Fleet request?
Answer:  Yes, you can tag a Spot Fleet request when you create it. However, the Spot Fleet itself cannot be tagged.
Context: ['You can request to launch Spot Instances with tags via Spot Fleet. The Fleet by itself cannot be tagged.']
Relevance Score: 5
Feedback: The response directly addresses the question and is consistent with the provided context. The context states that Spot Fleet cannot be tagged, and the response confirms this by stat
ing that you can tag a Spot Fleet request when you create it, but the Spot Fleet itself cannot be tagged.
==================================================

Question: Q: Where can I learn more about EFS?
Answer:
    You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.
Context: ['You can visit the Amazon EFS FAQ pageAmazon EFS FAQ page.']
Relevance Score: 4
Feedback: The response directly addresses the question by suggesting a place to learn more about EFS. The context supports this by explicitly mentioning the Amazon EFS FAQ page. However, the
response could have been more comprehensive by explaining what the FAQ page is or providing additional resources.
==================================================
```

In [ ]:
```python
# calculating the overall Relevance Scores of the ChatBot
relevance_scores['relevance_score'] = relevance_scores['relevance_score'].astype(int)

print("Overall Relevance Scores Rating of the ChatBot:"
f" {round(np.mean(relevance_scores['relevance_score']), 2)}")
```

```
Overall Relevance Scores Rating of the ChatBot: 4.54
```

In [ ]:
```python
# Saving to relevance scores .csv file
relevance_scores.to_csv('./scores/relevance_scores.csv', index=False)
```

Answer Correctness

Answer Correctness answers the question: **Is the response pertinent to the question and context?**

Answer correctness is evaluated based on how accurately the response matches the target. Scores reflect the degree of semantic similarity and factual correctness, with higher scores indicating greater accuracy.

Scoring Criteria

- *Score 1:* The response is completely incorrect. It is completely different from or contradicts the provided target.
- *Score 2:* The response demonstrates some degree of semantic similarity and includes partially correct information. However, the response still has significant discrepancies with the provided target or inaccuracies."
- *Score 3:* The response addresses a couple of aspects of the input accurately, aligning with the provided target. However, there are still omissions or minor inaccuracies.
- *Score 4:* The response is mostly correct. It provides mostly accurate information, but there may be one or more minor omissions or inaccuracies.
- *Score 5:* The response is correct. It demonstrates a high degree of accuracy and semantic similarity to the target.

In [ ]:
```python
from metrics.answer_correctness import AnswerCorrectness

answer_correctness = AnswerCorrectness()
answer_correctness_scores = answer_correctness.evaluate(chatbot_response_df)
```

```
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-git-credential` if using via `huggingface-cli` if you want
to set the git credential as well.
Token is valid (permission: write).
Your token has been saved to C:\Users\tapas\.cache\huggingface\token
Login successful
13it [02:00,  9.26s/it]
```

Answer Correctness result

In [ ]:
```python
# Restrincting to only 2 output for the sake of readability
for row in answer_correctness_scores.head(2).iterrows():
    print(f"Question: {row[1]['question']}")
    print(f"Answer: {row[1]['generated_answer']}")
    print(f"Context: {row[1]['context']}")
    print(f"Correctness Score: {row[1]['correctness_score']}")
    print(f"{row[1]['correctness_score_feedback']}")
    print('==================================================\n')
```

```
Question: Q. Can I tag a Spot Fleet request?
Answer:  Yes, you can tag a Spot Fleet request when you create it. However, the Spot Fleet itself cannot be tagged.
Context: ['You can request to launch Spot Instances with tags via Spot Fleet. The Fleet by itself cannot be tagged.']
Correctness Score: 4
Feedback: The response correctly explains that a Spot Fleet request can be tagged during creation, and clarifies that the Spot Fleet itself cannot be tagged, which is in line with the target.
However, the response could have been more detailed in explaining the concept of tagging Spot Fleet requests.
==================================================

Question: Q: Where can I learn more about EFS?
Answer:
    You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.
Context: ['You can visit the Amazon EFS FAQ pageAmazon EFS FAQ page.']
Correctness Score: 4
Feedback: The response is relevant to the given question and it includes the correct information. It suggests visiting the Amazon EFS FAQ page which aligns with the provided target. However,
it doesn't directly address the 'learn more' part of the question.
==================================================
```

In [ ]:
```python
# calculating the overall Faithfullness Score of the ChatBot
answer_correctness_scores['correctness_score'] = answer_correctness_scores['correctness_score'].astype(int)

print("Overall Correctness Score Rating of the ChatBot:"
f" {round(np.mean(answer_correctness_scores['correctness_score']), 2)}")
```

```
Overall Correctness Score Rating of the ChatBot: 3.85
```

In [ ]:
```python
# Saving to answer correctness scores .csv file
answer_correctness_scores.to_csv('./scores/answer_correctness_scores.csv', index=False)
```

Context Evaluation

To ensure the accuracy and relevance of the retrieved contexts in the ChatBot responses, I have use the **Entity Recall** metric.

Entity Recall

Entity Recall measures the system's ability to correctly recall all relevant entities within the context compared to a set of reference entities. This metric evaluates whether the system can identify and retrieve all key entities necessary for a comprehensive understanding of the query context.

Methodology

- **Extract Entities**: Use Named Entity Recognition (NER) tools such as **SpaCy** and **NLTK** to dynamically extract entities from both the retrieved context and the generated answer.
- **Compare Entities**: Compare the extracted entities from the retrieved context with those from the generated answer.
- **Calculate Entity Recall**: Calculate the recall as the number of correctly recalled entities divided by the total number of relevant entities in the generated answer.

In [ ]:
```python
import spacy
nlp = spacy.load("en_core_web_sm")

def extract_entities(text):
    doc = nlp(text)
    return [ent.text for ent in doc.ents]

def entity_recall(true_entities, retrieved_entities):
    true_positive = len(set(true_entities) & set(retrieved_entities))
    recall = true_positive / len(true_entities) if true_entities else 0
    return recall
```

In [ ]:
```python
entity_recall_output = []
for row in tqdm(chatbot_response_df.iterrows()):
    generated_answer_entities = extract_entities(row[1]['generated_answer'])
    context_entities = extract_entities(row[1]['context'][0])

    entity_recall_output.append(entity_recall(context_entities, generated_answer_entities))

entity_recall_scores = pd.concat([pd.DataFrame({'entity_recall_score': entity_recall_output}), chatbot_response_df], axis=1)
```

```
13it [00:01, 12.27it/s]
```

Entity Recall result

In [ ]:
```python
# Restrincting to only 2 output for the sake of readability
for row in entity_recall_scores.head(2).iterrows():
    print(f"Question: {row[1]['question']}")
    print(f"Answer: {row[1]['generated_answer']}")
    print(f"Context: {row[1]['context']}")
    print(f"Entity Recall Score: {round(row[1]['entity_recall_score'], 2)}")
    print('==================================================\n')
```

```
Question: Q. Can I tag a Spot Fleet request?
Answer:  Yes, you can tag a Spot Fleet request when you create it. However, the Spot Fleet itself cannot be tagged.
Context: ['You can request to launch Spot Instances with tags via Spot Fleet. The Fleet by itself cannot be tagged.']
Entity Recall Score: 0.5
==================================================

Question: Q: Where can I learn more about EFS?
Answer:
    You can learn more about Amazon EFS by visiting the Amazon EFS FAQ page.
Context: ['You can visit the Amazon EFS FAQ pageAmazon EFS FAQ page.']
Entity Recall Score: 0.67
==================================================
```

In [ ]:
```python
# calculating the overall Faithfullness Score of the ChatBot
entity_recall_scores['entity_recall_score'] = entity_recall_scores['entity_recall_score'].astype(int)

print("Overall Entity Recall Score of the ChatBot:"
f" {round(np.mean(entity_recall_scores['entity_recall_score']), 2)}")
```

```
Overall Entity Recall Score of the ChatBot: 0.08
```

In [ ]:
```python
# Saving to answer correctness scores .csv file
entity_recall_scores.to_csv('./scores/entity_recall_scores.csv', index=False)
```

Methods for Improving ChatBot response

Currently, the ChatBot performs well and provides relevant responses based on the FAQs. However, several improvements can be made to enhance its response quality. These improvements are as follows:

1. **Improved Chunking and Text Splitting**
   - *Current Approach*: The entire answer is converted to embeddings and stored in the vector store.
   - *Improvement*: Utilize more sophisticated chunking and text splitting techniques to ensure meaningful segments are created for better embedding and retrieval.
2. **Use Different Vector Stores for Context Retrieval**
   - *Current Approach*: The ChatBot currently uses Chroma DB.
   - *Improvement*: Experiment with different vector stores such as Pinecone, Weaviate, FAISS, etc., to optimize context retrieval and potentially improve performance.
3. **User Feedback on the Response**
   - *Current Approach*: No feedback mechanism is in place.
   - *Improvement*: Integrate a feedback module to record users' feedback on the generated responses, allowing for continuous learning and improvement based on real user interactions.
4. **Using Different LLMs for Better Response Generation and Evaluation Metrics**
   - *Current Approach*: The ChatBot uses Mixtral-8x7B-Instruct-v0.1.
   - *Improvement*: Experiment with different models such as Meta's LLAMA, Google's Gemma, and OpenAI's models to enhance response generation and evaluation metrics.
5. **Improved Irrelevant Query Handling**
   - *Current Approach*: There is no specific mechanism for handling out-of-context queries.
   - *Improvement*: Implement a separate LLM specifically for filtering out and processing out-of-context queries to ensure the ChatBot handles irrelevant queries more effectively.