



## **Senior ML Engineer Tech Assessment**

**By:** Maria Camila Ortiz

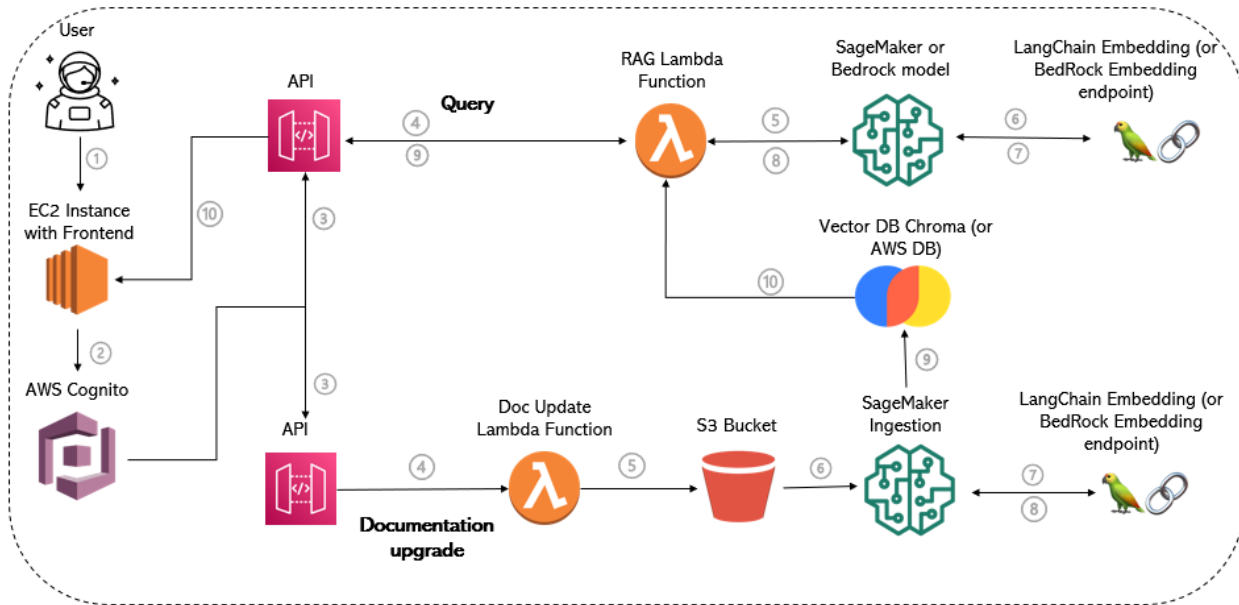
### **ALPACA-BOT Solution**

Scenario Company X has a large amount of documentation that their developers need to navigate. They have noticed that their developers often spend significant amounts of time searching through documentation or asking other developers simple questions that are in the documentation. The Company has reached out to Loka to assist with building a tool to address this issue. After some discussion, it is agreed that the first step of the collaboration should be a POC whose goal would be to prove that the system would significantly shorten the amount of time developers spend looking through documentation. The POC will cover only a subset of their data and will initially just be applied to one of the teams. Upon further investigation, they mention that their main goal would be to have a system that could assist developers with parts of the documentation they aren't familiar with, as in these cases they typically reach out to coworkers with some pretty simple questions. This has several issues, among them experienced members have their work interrupted often and responses are sometimes based on old information, as documentation is often updated, causing problems down the line. They would also like for the system to be able to point the user to further reading, by pointing them towards the source for the response and towards other documents that may be relevant to what they are currently working on. However, this last request is a nice-to-have and not a mandatory feature. They can accept this being implemented later. The documentation provided for the POC is public (it's AWS documentation) and as such has no limitations on usage. However, the final system will also handle internal documentation that contains sensitive information that has proprietary (can't be shared or accessed externally) and geographical restrictions (can't leave the US). They also provided some example questions they would like the system to be able to respond to for the POC:

- What is SageMaker?
- What are all AWS regions where SageMaker is available?
- How to check if an endpoint is KMS encrypted?
- What are SageMaker Geospatial capabilities?

Your task is to design the overall solution for Company X. Determine what parts of that solution should be part of the POC and begin implementing the POC.

## ALPACA-BOT ARCHITECTURE POC



### Assumptions:

- The user lacks access to the S3 bucket for data updates.
- Manual data updates are feasible through user-driven uploads via the application interface.

### WORKFLOW:

1. The user accesses the app.
2. The user logs in with an AWS Cognito user.
3. The user chooses between asking a question or updating a file.
  - a. **Documentation Update:**
    4. A Lambda function is triggered.
    5. The file is updated in the S3 bucket.
    6. The new data is ingested into a SageMaker notebook.
    7. The notebook employs an embedding model, specifically from LangChain (an openAI source).
    8. The embedding model processes the ingested data, returning it as vectorized data.
    9. The vectorized data is stored in a Vector Database, such as Chroma (used for a limited dataset). This option can be substituted with an AWS Vector Database.
    10. The vectorized data is sent to the RAG activation Lambda and ingested into the model.
  - b. **Query:**
    4. The RAG Lambda Function is triggered with the user's query.

5. The query is processed within the model.
6. An embedding process is applied to the query.
7. The model processes the vectorized query and generates an answer.
8. The answer is sent to the Lambda function.
9. The Lambda function connects to the API.
10. The API responds with the answer, which is displayed in the application interface.

#### **Future Enhancements:**

##### **Authentication Process with AWS Cognito:**

1. In the authentication process with AWS Cognito, an AWS CloudFront implementation can be introduced to monitor the geographic zone from which the user is accessing the data. This implementation enables the restriction of replies based on the user's geographical location.

##### **SageMaker Model Improvement:**

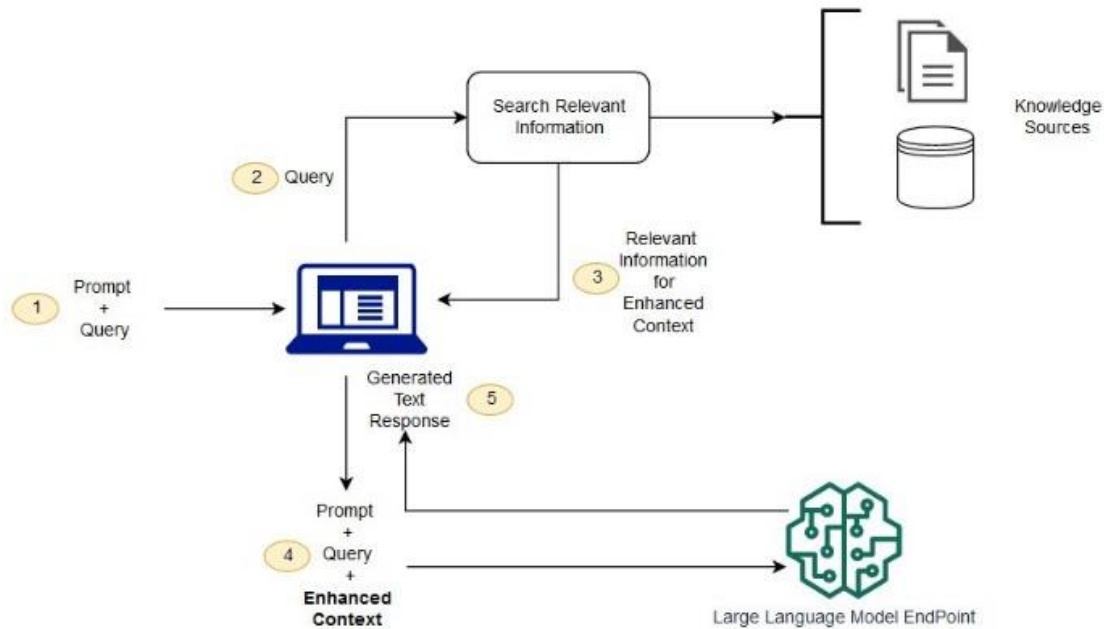
2. Within the SageMaker model, consider integrating a KNN (K-Nearest Neighbors) means model. This addition enhances the reply mechanism by returning the top X documents from the source documentation that contain information related to the user's query.

## **MISTRAL 7B - RAG**

According to HuggingFace LLM Leaderboard, Mistral 7B has surpassed Llama in some of its versions, which is quite surprising considering it is a smaller model. It is an open-source model with excellent documentation, making relevant information easily accessible. Furthermore, as it was not possible to test the endpoint on SageMaker, a small model was necessary for conducting tests, parameter adjustments, and the RAG process locally.

To deploy the model, [ollama.ai](https://ollama.ai), an open-source tool enabling straightforward modifications to the utilized models, was utilized. Moreover, it features an interface that facilitates the addition of different users.

To address the business's ongoing need for finding and chatting with updated documentation, RAG was employed. Retrieval Augmented Generation or RAG is used to retrieve data from outside a foundation model and augment the prompts by adding the relevant retrieved data in context. As foundation models are usually trained offline, they do not reply with updated data. Additionally, FM are trained on very general domain corpora, making them less effective for domain-specific tasks.



From <https://aws.amazon.com/es/what-is/retrieval-augmented-generation/>

In this image, the process is described: upon receiving a prompt + query, information is sought within the context (information sources), which is then processed within the model to provide a text response.

These are some of the results provided applying RAG into MISTRAL 7B, with the source documents provided in the test:

```
> Question:
What is SageMaker?
Amazon SageMaker is a fully managed platform that provides deep learning functionalities as a service, allowing developers and data scientists to build, train, and deploy machine learning models quickly. It offers pre-built algorithms and supports custom algorithms as well, enabling users to bring their own data and models. For more information on creating model packages in the SageMaker console, refer to the provided context.
```

```

> Question:
How to check if an endpoint is KMS encrypted?
To check if an Amazon SageMaker endpoint uses a KMS key for encryption, you can examine the
endpoint configuration in the AWS Management Console or by using the AWS CLI. Here's how to do
it in both ways:

1. **AWS Management Console:**
  - Sign in to the AWS Management Console and navigate to the Amazon SageMaker dashboard.
  - Select the endpoint you want to check. In the "Endpoint configurations" tab, look for the
    "Encryption configuration" section. If it shows a value for either the Key ID or Key ARN under
    "KMS key ID," then the endpoint is encrypted using KMS.

2. **AWS CLI:**
  - First, make sure you have the AWS CLI installed and configured with the necessary access
    keys.
  - Run the following command to list all endpoints and their encryption configurations:
    ```
    aws sagemaker describe-endpoint --endpoint-name <your_endpoint_name>
    ```
  - Look for the `EncryptionConfiguration` object in the output. If it has a value for either
    `KeyId` or `KeyArn`, then the endpoint is encrypted using KMS.

> source_documents\aws-properties-sagemaker-endpointconfig-datacaptureconfig.txt:
The AWS Key Management Service (AWS KMS) key that Amazon SageMaker uses to encrypt the captured
data at rest using Amazon S3 server-side encryption. The KmsKeyId can be any of the following

```

## Resources on Retrieval-Augmented Generation (RAG) and Hugging Face & AWS Language Models

1. Hugging Face Model Spaces Leaderboard. (<https://huggingface.co/spaces/mteb/leaderboard>)
2. Amazon Web Services (AWS) - Retrieval Augmented Generation. (<https://aws.amazon.com/es/what-is/retrieval-augmented-generation/>)
3. Ollama - Private GPT Chat with Docs. (<https://github.com/PromptEngineer48/Ollama/tree/main/2-ollama-privateGPT-chat-with-docs>)
4. Skim AI - Mistral 7B vs Llama2: Las 5 Diferencias Clave Entre los Principales LLM de Código Abierto. (<https://skimai.com/es/mistral-7b-vs-llama2-las-5-diferencias-clave-entre-los-principales-llm-de-codigo-abierto/>)
5. Hugging Face Demos - Building a Simple RAG Chatbot. (<https://gitlab.com/juliensimon/huggingface-demos/-/blob/main/langchain/rag-demo-sagemaker-textract/Building%20a%20simple%20RAG%20chatbot.ipynb>)
6. Amazon SageMaker Examples - JumpStart Question Answering. ([https://github.com/aws/amazon-sagemaker-examples/blob/main/introduction\\_to\\_amazon\\_algorithms/jumpstart\\_question\\_answering/Amazon\\_JumpStart\\_Question\\_Answering.ipynb](https://github.com/aws/amazon-sagemaker-examples/blob/main/introduction_to_amazon_algorithms/jumpstart_question_answering/Amazon_JumpStart_Question_Answering.ipynb))