Part 2: Interaction with Azure OpenAI y Dockerization

In this challenge, you will immerse yourself in the world of natural language processing (NLP) and interacting with external APIs. You will use the AzureOpenAI API to perform a specific task related to text. Additionally, to ensure that your solution is easily deployable and scalable, we will ask you to dockerize your application.

2.1 Objective

Your task is to create a small Python application that uses the Azure OpenAI API to generate a summary of a set of provided articles or texts. This summary should capture the key points of the input texts in a coherent and concise manner. For this you will have to create a prompt so that the model generates the summary in the best possible way.

Be creative and ask the model using the prompt for more complex things apart from the summary of the text, such as a list of keywords, references to other texts, ... Whatever you come up with to make the summarization more complete.

2.2 Guidelines

1. Texts: You can use texts of your choice for this task. These can be news articles, blogs, white papers, etc. The only condition is that the document must be extensive (minimum 3 pages of text).

2. Interaction with Azure OpenAI: You will use the provided endpoint to send the text to the Azure OpenAI API and receive the generated summary. To create the connection with the API you will use the library Semantic-Kernel.

In the link: https://github.com/microsoft/semantic-kernel/blob/main/python/notebooks/04-kernel-arguments-chat.ipynb you can find a notebook with examples on how to use this library.

In this case you will have to use the service AzureOpneAI (The following image is part of the notbook from the link):

```
elif selectedService == Service.AzureOpenAI:
    from semantic_kernel.connectors.ai.open_ai import AzureChatCompletion

    deployment, api_key, endpoint = sk.azure_openai_settings_from_dot_env()
    service_id = "aoai_chat_completion"
    kernel.add_service(
        AzureChatCompletion(service_id=service_id, deployment_name=deployment, endpoint=endpoint, api_key=api_key
    )
```

We will provide the variables deployment, api_key and enpoint.

2.3. Dockerization of the APP:

Once you have your application up and running, the next step is to dockerize it. Create a “Dockerfile” that specifies how your application should be built and run it in a container. The docker must install the requirements.txt file and then run the python file that will call the API to generate the summary. Once the summary is created you can decide how to show it to the user (Terminal, creating a document, ...)

2.4 What will you send us?

We will only need a link to the repository in GitHub where you made the commits of the app. Please be sure to add in the repo the Dockerfile, the python files, the requirements and the README.

2.5 Extra

Any functionality developed that can add value to the application will be considered a plus. For example, the summary text generated by the model has several sections or the summary is shown to the user through a webapp.

Be as creative as you want, it's your APP!