Set Up a Simple Gradio Interface to Interact with Your Models

Estimated time needed: 30 minutes

Imagine you're developing a customer support chatbot for a company. The goal is to provide a seamless and interactive way for customers to get answers to their questions. To achieve this, you need an interface where users can input their queries and receive responses generated by an large language model (LLM). This lab will guide you through creating such an interface using Gradio. You'll learn how to integrate various Gradio components, like text input fields, buttons, and display elements, to create an intuitive and user-friendly experience.

Creating an interface is crucial for several reasons:

- User accessibility: A front-end interface makes it easy for users to interact with the LLM without needing technical expertise.
- Enhanced user experience: An intuitive interface can provide a better user experience, making interactions smoother and more efficient.
- Customization: Gradio allows you to customize the interface to meet specific needs, whether for a chatbot, a data analysis tool, or other applications.
- Seamless integration: Gradio's flexibility enables you to seamlessly integrate the front-end with various backend technologies, including LLMs.

Source: DALL-E

In this lab, you will learn how to use Gradio to set up a front-end interface that allows users to interact with a backend LLM. Gradio is a powerful and user-friendly library for creating customizable web-based interfaces. By the end of this lab, you will have the skills to use essential and commonly used Gradio elements to build an interface that communicates with an LLM, enabling you to create a functional chatbot.

This lab will provide you with hands-on experience in building an interactive interface, empowering you to create applications that leverage the power of LLMs through user-friendly front-end solutions.

Learning objectives

By the end of this project, you will be able to:

- **Use Gradio to build interactive front-end interfaces**, enabling users to interact with backend LLMs seamlessly
- **Create a functional chatbot**, allowing users to input queries and receive responses from an LLM
- **Implement essential and commonly used Gradio elements**, such as text input fields, buttons, and display areas, to enhance user experience
- **Customize and deploy web-based applications**, facilitating various use cases like customer support, data analysis, and more

Setup

Setting up a virtual environment

Let's create a virtual environment. Using a virtual environment allows you to manage dependencies for different projects separately, avoiding conflicts between package versions.

In the terminal of your Cloud IDE, ensure that you are in the path `/home/project`, then run the following commands to create a Python virtual environment.

```
pip install virtualenv
virtualenv my_env # create a virtual environment named my_env
source my_env/bin/activate # activate my_env
```

Installing necessary libraries

To ensure a seamless execution of our scripts, and considering that certain functions within these scripts rely on external libraries, it's essential to install some prerequisite libraries before you begin. For this project, the key libraries you will need are `Gradio` for creating user-friendly web interfaces and `IBM Watsonx AI` for leveraging advanced LLM model from IBM watsonx's API.

- **`gradio`** allows you to build interactive web applications quickly, making your AI models accessible to users with ease.
- **`ibm-watsonx-ai`** for using LLMs from IBM's watsonx.ai.
- **`langchain`, `langchain-ibm`, `langchain-community`** for using relevant features from LangChain.

Here's how to install these packages (still from your terminal):

```
# installing necessary pacakges in my_env
python3.11 -m pip install \
gradio==4.44.0 \
ibm-watsonx-ai==1.1.2 \
langchain==0.2.11 \
langchain-community==0.2.10 \
langchain-ibm==0.1.11
```

Now, the environment is ready to create Python files.

Quick tutorial to Gradio

Gradio overview

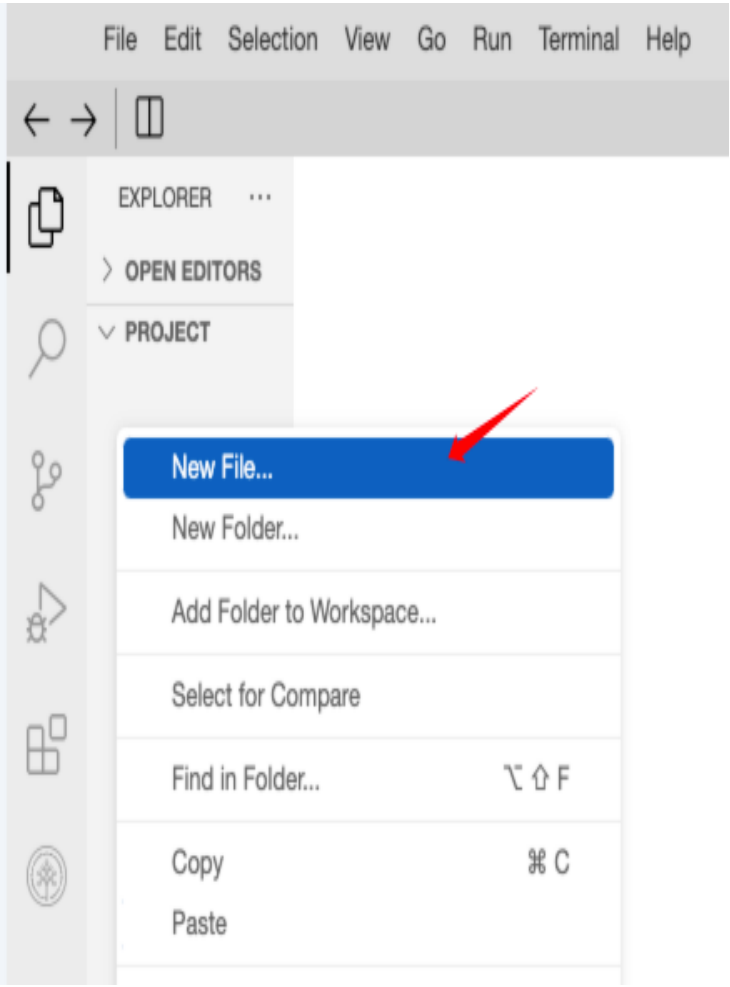
Gradio can be used to create a web interface to seamlessly bridge the gap between machine learning models and end-user interaction. It enables developers to effortlessly transform their algorithms into accessible, interactive applications. With Gradio, you can quickly design web interfaces that allow users to input data in various forms—such as text, images, or audio—and instantly view the output generated by your model. For more information, please visit Gradio.

In this project, your LLM-based resume advisor application will need a web interface to interact with. So, let's get familiar with Gradio and create a simple Gradio application first.

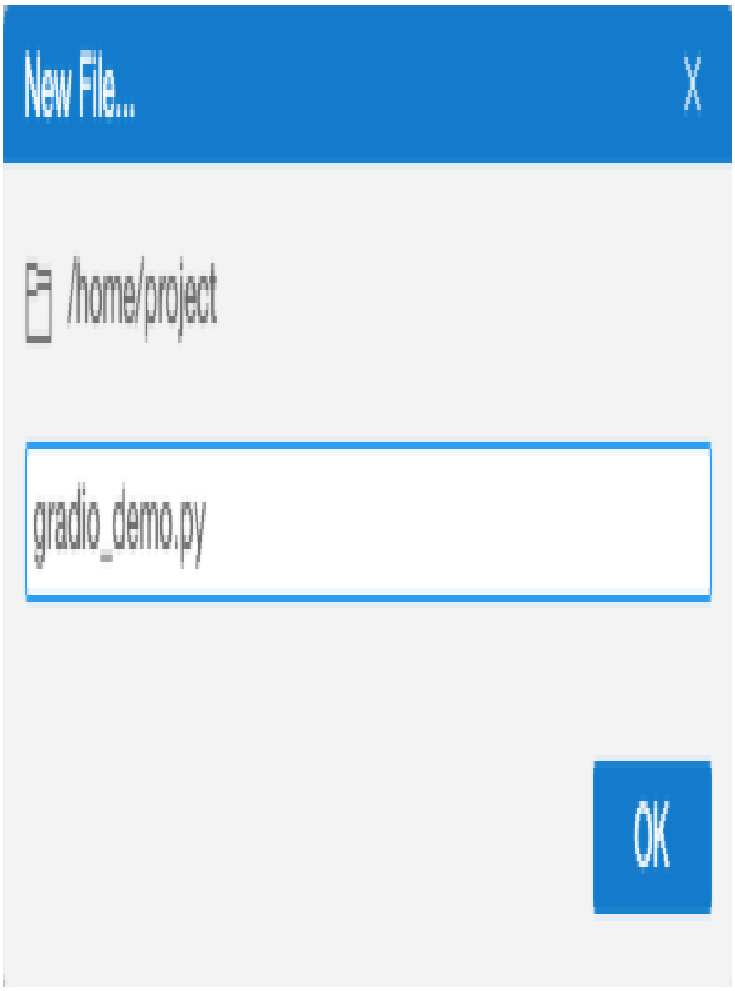
Create a Gradio demo

Create a .py file

First, you need to create a .py file in cloud IDE on your left. Under the `EXPLORER` tab, open the `PROJECT` directory, right click and choose `New file`.



Then, you can name your file `gradio_demo.py`.



A demo of sum calculator

You will create an application that can calculate the sum of your input numbers.

In the `gradio_demo.py`, you can enter the following code.

The `gr.Number()` element from gradio is being used to create a numeric field for the user to enter numbers as input or display numeric output.

```
import gradio as gr
def add_numbers(Num1, Num2):
    return Num1 + Num2
# Define the interface
demo = gr.Interface(
    fn=add_numbers,
    inputs=[gr.Number(), gr.Number()], # Create two numerical input fields where users can enter numbers
    outputs=gr.Number() # Create numerical output fields
)
# Launch the interface
demo.launch(server_name="127.0.0.1", server_port= 7860)
```

Launch the application

Return to the terminal and verify that the virtual environment `my_env` label appears at the start of the line, which means that you are in the `my_env` environment that you just created. Next, execute the following command to run the Python script.

```
python3.11 gradio_demo.py
```

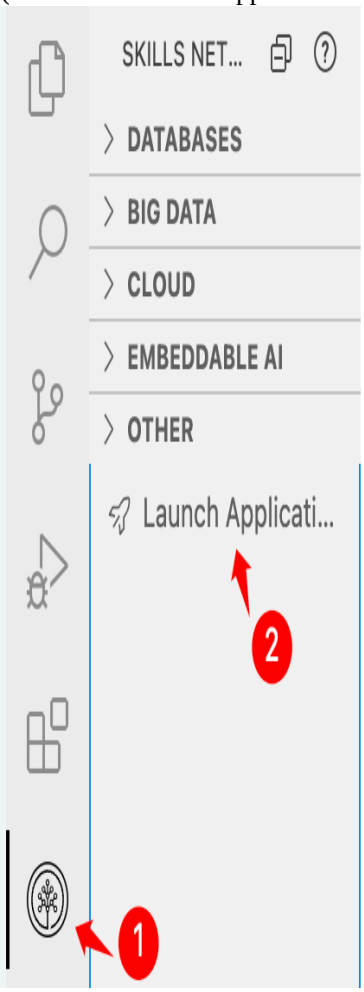
After it runs successfully, you will see a message similar to following in the terminal:

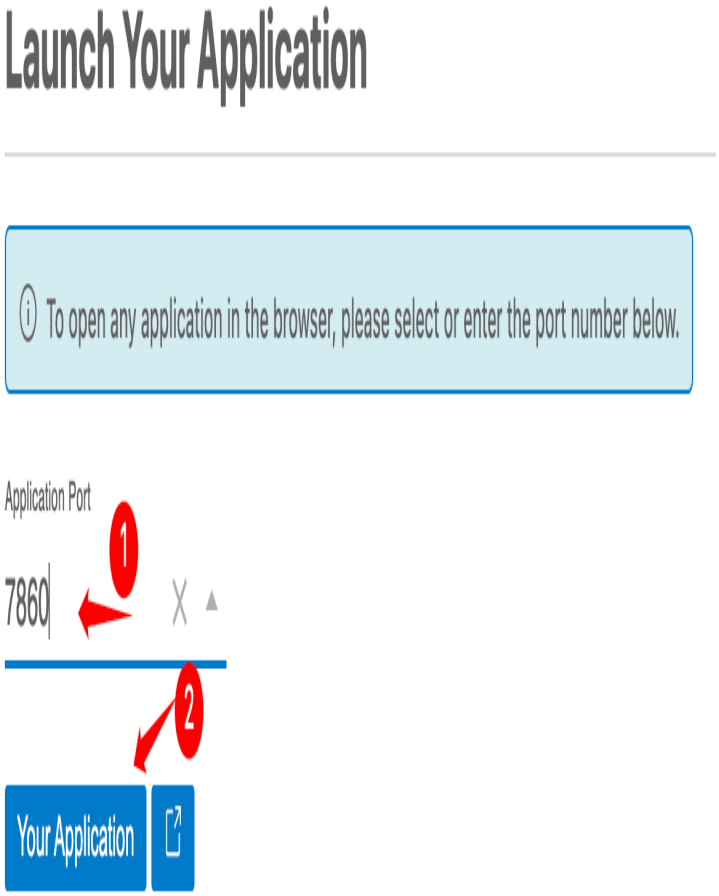
```
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

Since the web application is hosted locally on port 7860, click on the following button to view the application we've developed.

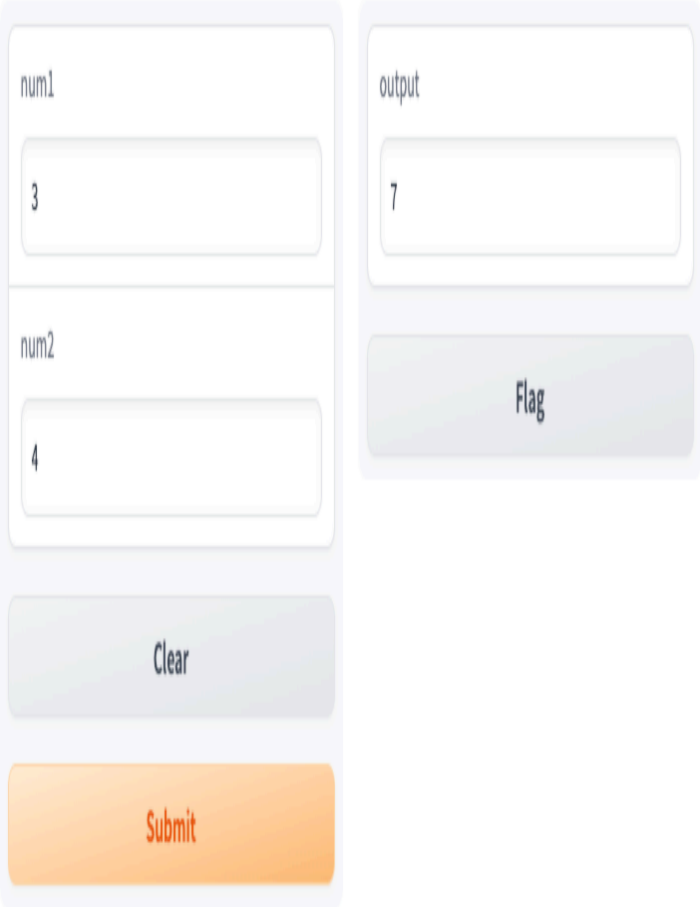
Web Application

(Note: if this “Web Application” button does not work, follow the following picture instructions to launch the app.”)





Let's take a look at the web application. It shows an example of sum of 3 and 4. You're encouraged to experiment with the web app's inputs and outputs!



If you want to stop the script, you can press `Ctrl+C` in the terminal and close the appliction window.

Exercise

Can you create a Gradio application that can combine two input sentences together? Take your time to complete this exercise.

▶ Let's take a look of the answer.

You just had a first taste of the Gradio interface; it's easy right? If you want to learn more about customization in Gradio, you can take this guided project **Bring your Machine Learning model to life with Gradio**. You can find more relevant courses and projects on cognitiveclass.ai

For the rest of this project, you will use Gradio as an interface for the created application.

Quick start on watsonx.ai LLM

watsonx.ai introduction

watsonx.ai is IBM's commercial generative AI and scientific data platform based on cloud technology. It encompasses a studio, data store, and governance toolkit, designed to support multiple LLMs. This platform is tailored for a wide range of AI development tasks, offering developers access to IBM's own series of LLMs as well as models from Meta's Llama-3 and Mixtral model.

In this section, you will be guided through the process of using watsonx.ai's API to create a simple Q&A bot. This bot will leverage the advanced capabilities of watsonx.ai to understand and respond to user queries accurately. Whether you're new to programming or an experienced developer, this step-by-step tutorial will equip you with the knowledge to integrate watsonx.ai's LLMs into your applications.

Create a Q&A bot

This tutorial will walk you through the process of creating a Q&A chatbot leveraging the `mistralai/mixtral-8x7b-instruct-v01` model developed by Mistral AI. This powerful foundation model has been seamlessly integrated into IBM's watsonx.ai platform, simplifying your development journey. The provided API eliminates the need for generating complex API tokens, streamlining the application creation process. The `mistralai/mixtral-8x7b-instruct-v01` model is equipped with the following features:

- Supports Q&A
- Summarization
- Classification
- Generation
- Extraction
- Retrieval-augmented generation
- Code generation

This model aligns perfectly with your objective for the application being created. For models and their IDs, please refer to here.

Note: The `mistralai/mixtral-8x7b-instruct-v01` model, like any AI technology, has its limitations, and it is possible to encounter nonsensical responses occasionally. The primary objective of this project is to provide guidence on utilizing LLMs with watsonx.ai.

Follow these step-by-step instructions to create your application:

1. Still in the `PROJECT` directory, create a new Python file named `simple_llm.py` (you are welcome to choose a different name if you prefer).

2. Enter the following script content into your newly created `simple_llm.py` file and save your changes. Line-by-line explanation of the code snippet is provided.

```
# Import necessary packages
from ibm_watsonx_ai.foundation_models import ModelInference
from ibm_watsonx_ai.metanames import GenTextParamsMetaNames as GenParams
from ibm_watsonx_ai import Credentials
from langchain_ibm import WatsonxLLM
# Model and project settings
model_id = 'mistralai/mixtral-8x7b-instruct-v01' # Directly specifying the model
# Set necessary parameters
parameters = {
    GenParams.MAX_NEW_TOKENS: 256,  # Specifying the max tokens you want to generate
    GenParams.TEMPERATURE: 0.5, # This randomness or creativity of the model's responses
}
project_id = "skills-network"
# Wrap up the model into WatsonxLLM inference
watsonx_llm = WatsonxLLM(
    model_id=model_id,
    url="https://us-south.ml.cloud.ibm.com",
    project_id=project_id,
    params=parameters,
)
# Get the query from the user input
query = input("Please enter your query: ")
# Print the generated response
print(watsonx_llm.invoke(query))
```

3. Open your terminal and ensure that you are operating within the virtual environment (`my_env`) you previously established.

4. Run the following command in the terminal.

```
python3.11 simple_llm.py
```

Upon successful execution of the code, you can input our query in the terminal. A response is then generated.

The following picture is showing an example response based on the query: `How to be a good data scientist?`.

```
(my_env) theia@theia-kangwang:/home/project$ python3.11 simple_llm.py
Please enter your query: How to be a good data scientist?



Data Scientists are the new stars of the tech world. They are in high demand and are we
ll paid. But what does it take to be a good data scientist?

First, you need to have a strong foundation in mathematics and statistics. You should b
e comfortable with concepts like probability, linear algebra, and calculus. You should
also be familiar with statistical methods like regression analysis, hypothesis testing,
 and experimental design.

Second, you need to have programming skills. Python is the most popular language for da
ta science, but R, SQL, and Java are also commonly used. You should be able to write cl
ean, efficient code and be familiar with data structures, algorithms, and software engi
neering principles.

Third, you need to have a deep understanding of the domain you are working in. This mea
ns knowing the business problems you are trying to solve, the data you are working with
, and the tools and techniques that are commonly used in the field.

Fourth, you need to be able to communicate effectively. Data scientists need to be able
 to explain complex concepts to non-technical stakeholders, present findings in a clear
 and concise manner, and work well in a team.

Fifth, you need
```

In the code, you simply used `"skills-network"` as `project_id` to gain immediate, free access to the API without the need for initial registration. It's important to note that this access method is exclusive to this cloud IDE environment. If you are interested in using the model/API in a local environment, detailed instructions and further information are available in this tutorial.

Gradio and LLM

Integrate the application into Gradio

Having successfully created a Q&A bot with your script, you might notice that responses are only displayed in the terminal. You may wonder if it's possible to integrate this application with Gradio to leverage a web interface for inputting questions and receiving responses.

The following code guides you through this integration process. It includes three components:

- Initializing the model
- Defining the function that generates responses from the LLM
- Constructing the Gradio interface, enabling interaction with the LLM
    - The `gr.Textbox` element is being used to create text field and hold users' input query and LLM's output.

```
# Import necessary packages
from ibm_watsonx_ai.foundation_models import ModelInference
from ibm_watsonx_ai.metanames import GenTextParamsMetaNames as GenParams
from ibm_watsonx_ai import Credentials
from langchain_ibm import WatsonxLLM
import gradio as gr
# Model and project settings
model_id = 'mistralai/mixtral-8x7b-instruct-v01' # Directly specifying the model
# Set necessary parameters
parameters = {
```

```
    GenParams.MAX_NEW_TOKENS: 256,  # Specifying the max tokens you want to generate
    GenParams.TEMPERATURE: 0.5, # This randomness or creativity of the model's responses
}
project_id = "skills-network"
# Wrap up the model into WatsonxLLM inference
watsonx_llm = WatsonxLLM(
    model_id=model_id,
    url="https://us-south.ml.cloud.ibm.com",
    project_id=project_id,
    params=parameters,
)
# Function to generate a response from the model
def generate_response(prompt_txt):
    generated_response = watsonx_llm.invoke(prompt_txt)
    return generated_response
# Create Gradio interface
chat_application = gr.Interface(
    fn=generate_response,
    allow_flagging="never",
    inputs=gr.Textbox(label="Input", lines=2, placeholder="Type your question here..."),
    outputs=gr.Textbox(label="Output"),
    title="Watsonx.ai Chatbot",
    description="Ask any question and the chatbot will try to answer."
)
# Launch the app
chat_application.launch(server_name="127.0.0.1", server_port= 7860)
```

1. Navigate to the `PROJECT` directory, right-click, and create a new file named `llm_chat.py`.
2. Input the script provided above into this new file.
3. Open your terminal and ensure you are within the `my_env` virtual environment.
4. Execute the following code in the terminal to run the application.

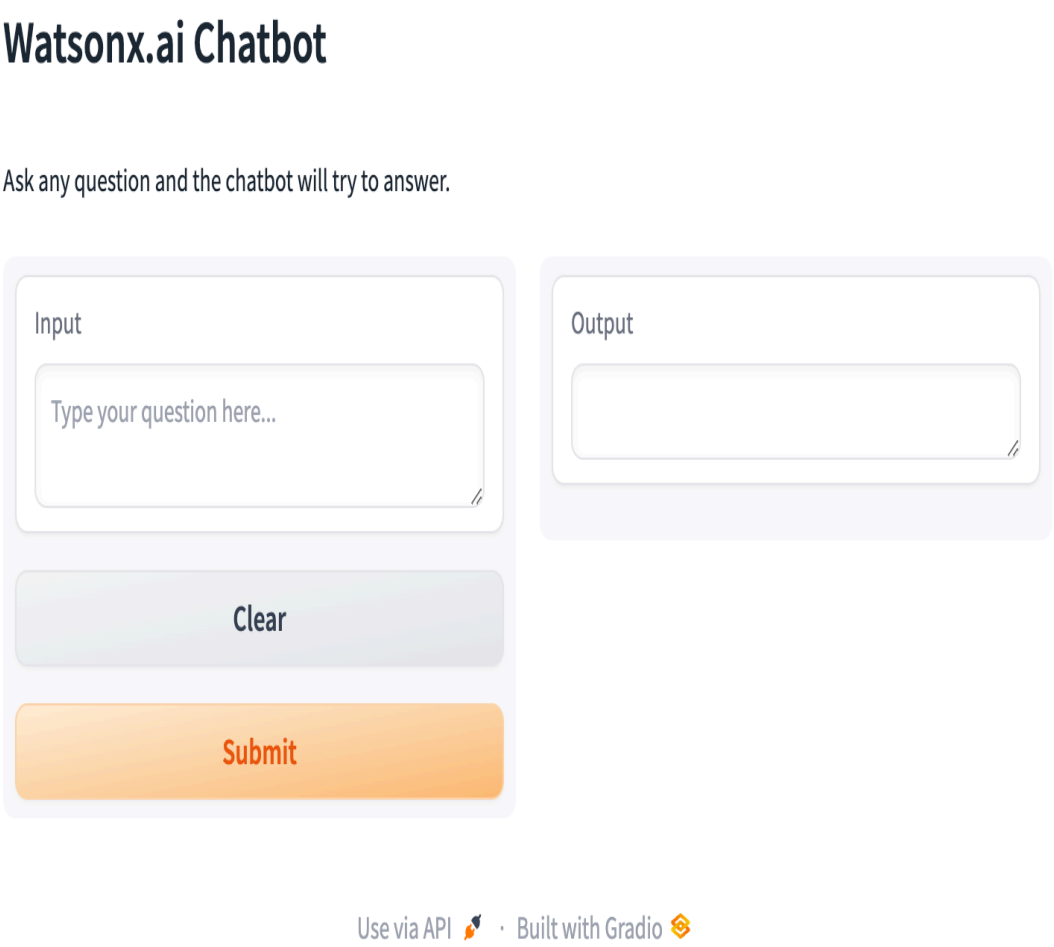
```
python3.11 llm_chat.py
```

After it has excuted successfully, you will see message similar to the following in the terminal:

5. Click the following button to launch and view the application.

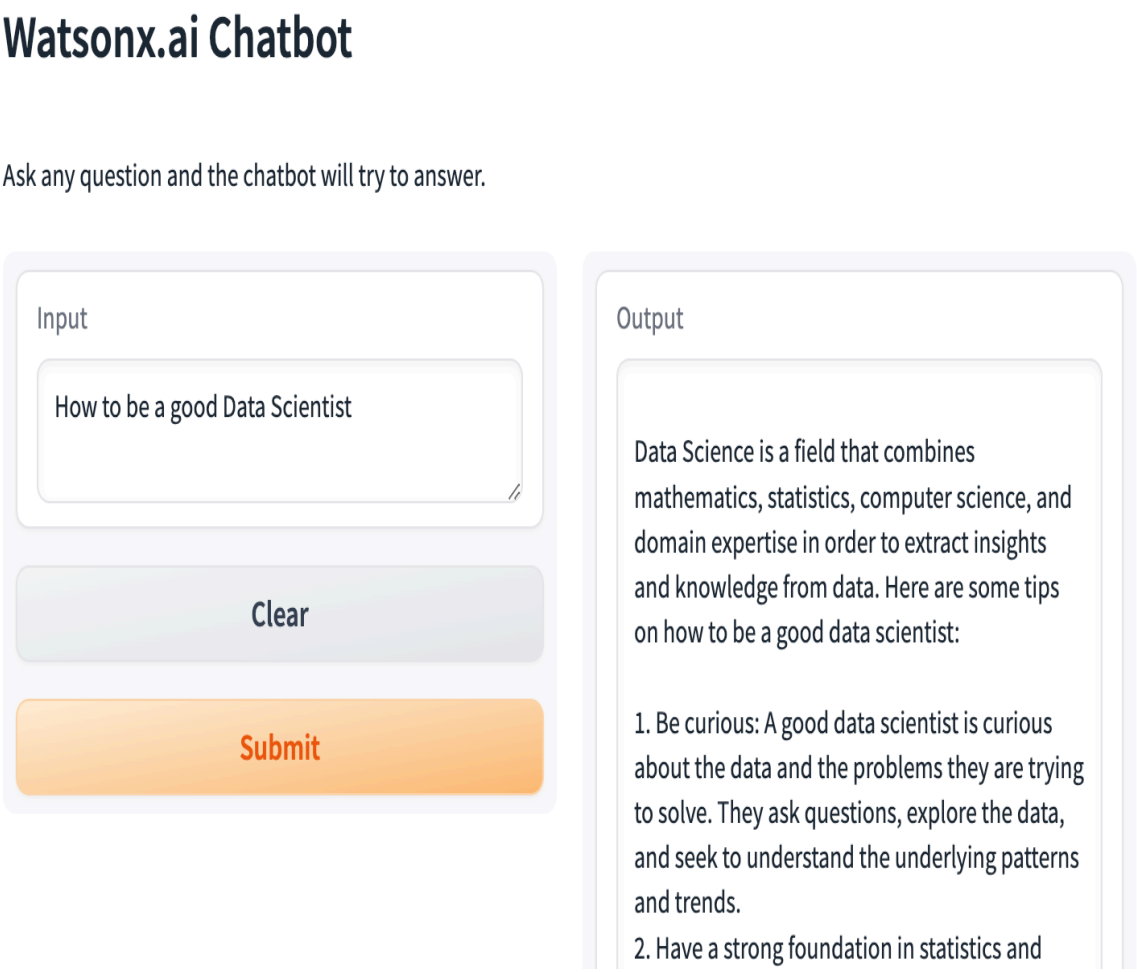
Web Application

The chatbot you have successfully created will be displayed, appearing as follows:



Now, feel free to ask any question to the chatbot.

Here is an example of the question asked:



(To terminate the script, press `Ctrl+C` in the terminal and close the appliction window.)

Exercise

You might observe that the responses from the LLM are occasionally incomplete. Could you identify the cause of this issue? Also, would you be able to modify the code to enable the model to generate more content?

Actually, all you need to do is:

▶ Click here for the answer

In the following section, we will explore how to develop a resume polisher using the knowledge we have just acquired.

Authors

Authors

Kang Wang

Kang Wang is a data scientist in IBM. He is also a PhD candidate in the University of Waterloo.

Other contributors

Joseph Santarcangelo

Joseph has a Ph.D. in electrical engineering, his research focused on using machine learning, signal processing, and computer vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Copyright © IBM Corporation. All rights reserved.