Prompt Notebook with Chat - Prompt Lab Notebook v1.1.0

This notebook contains steps and code to demonstrate inferencing of prompts generated in Prompt Lab in watsonx.ai with a chat format. It introduces Python API commands for authentication using API key and prompt inferencing using WML API.

Note: Notebook code generated using Prompt Lab will execute successfully. If code is modified or reordered, there is no guarantee it will successfully execute. For details, see: Saving your work in Prompt Lab as a notebook.

Some familiarity with Python is helpful. This notebook uses Python 3.10.

Notebook goals

The learning goals of this notebook are:

- Defining a Python function for obtaining credentials from the IBM Cloud personal API key
- Defining parameters of the Model object
- Using the Model object to generate response using the defined model id, parameters and the prompt input

Setup

watsonx API connection

This cell defines the credentials required to work with watsonx API for Foundation Model inferencing.

Action: Provide the IBM Cloud personal API key. For details, see documentation.

```
!pip install ibm_watsonx_ai
```

```
Requirement already satisfied: ibm_watsonx_ai in /usr/local/lib/python3.12/dist-packages (1.4.5)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (2.32.4)
Requirement already satisfied: httpx<0.29,>=0.27 in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (0.28.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (2.5.0)
Requirement already satisfied: pandas<2.3.0,>=0.24.2 in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (2.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (2025.10.5)
Requirement already satisfied: lomond in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (0.3.3)
Requirement already satisfied: tabulate in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (0.9.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (25.0)
Requirement already satisfied: ibm-cos-sdk<2.15.0,>=2.12.0 in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (2.1
Requirement already satisfied: cachetools in /usr/local/lib/python3.12/dist-packages (from ibm_watsonx_ai) (5.5.2)
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages (from httpx<0.29,>=0.27->ibm_watsonx_ai) (4.11.0
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<0.29,>=0.27->ibm_watsonx_ai)
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx<0.29,>=0.27->ibm_watsonx_ai) (3.11)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<0.29,>=0.27->ibm_
Requirement already satisfied: ibm-cos-sdk-core==2.14.3 in /usr/local/lib/python3.12/dist-packages (from ibm-cos-sdk<2.15.0,>=2.
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.14.3 in /usr/local/lib/python3.12/dist-packages (from ibm-cos-sdk<2.15.
Requirement already satisfied: jmespath<=1.0.1,>=0.10.0 in /usr/local/lib/python3.12/dist-packages (from ibm-cos-sdk<2.15.0,>=2.
Requirement already satisfied: python-dateutil<3.0.0,>=2.9.0 in /usr/local/lib/python3.12/dist-packages (from ibm-cos-sdk-core==
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas<2.3.0,>=0.24.2->ibm_watsonx
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<2.3.0,>=0.24.2->ibm_watsonx_
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<2.3.0,>=0.24.2->ibm_watson
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->ibm_watsonx_a
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.12/dist-packages (from lomond->ibm_watsonx_ai) (1.17.0)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packages (from anyio->httpx<0.29,>=0.27->ibm_watso
Requirement already satisfied: typing_extensions>=4.5 in /usr/local/lib/python3.12/dist-packages (from anyio->httpx<0.29,>=0.27-
```

```
!pip install git+https://github.com/ibm-granite-community/utils
```

```
Collecting git+https://github.com/ibm-granite-community/utils
  Cloning https://github.com/ibm-granite-community/utils to /tmp/pip-req-build-b_7574wn
  Running command git clone --filter=blob:none --quiet https://github.com/ibm-granite-community/utils /tmp/pip-req-build-b_7574w
  Resolved https://github.com/ibm-granite-community/utils to commit 3c725e243561d17b575a0a5496bfa3d3f28f6481
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.12/dist-packages (from ibm-granite-community-utils==0.1.d
Requirement already satisfied: langchain_core in /usr/local/lib/python3.12/dist-packages (from ibm-granite-community-utils==0.1.
Requirement already satisfied: typing_extensions in /usr/local/lib/python3.12/dist-packages (from ibm-granite-community-utils==0
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-gra
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/lib/python3.12/dist-packages (from langchain_core->
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-gra
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-granite
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-gr
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-grani
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/dist-packages (from jsonpatch<2.0.0,>=1.33.0->langc
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->langc
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->langcha
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.4
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->langch
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->lang
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->l
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->la
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4-
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.45
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.45-
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->langs
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.0.0->langsm
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.0.0->langsmith<1.
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packages (from anyio->httpx<1,>=0.23.0->langsmith<
```

```python
import os
from ibm_watsonx_ai import APIClient, Credentials
from ibm_granite_community.notebook_utils import get_env_var
import getpass

credentials = Credentials(
    url="https://us-south.ml.cloud.ibm.com",
    api_key=get_env_var("IBM_CLOUD_TOKEN")
)
```

```
IBM_CLOUD_TOKEN loaded from Google Colab secret.
```

Inferencing

This cell demonstrated how we can use the model object as well as the created access token to pair it with parameters and input string to obtain the response from the the selected foundation model.

Defining the model id

We need to specify model id that will be used for inferencing:

```python
model_id = "mistralai/mistral-medium-2505"
```

Defining the model parameters

We need to provide a set of model parameters that will influence the result:

```python
parameters = {
    "frequency_penalty": 0,
    "max_tokens": 2000,
    "presence_penalty": 0,
    "temperature": 0,
    "top_p": 1
}
```

Defining the project id or space id

The API requires project id or space id that provides the context for the call. We will obtain the id from the project or space in which this notebook runs:

```
project_id = get_env_var("PROJECT_ID")
space_id = get_env_var("SPACE_ID")
```

```
PROJECT_ID loaded from Google Colab secret.
SPACE_ID loaded from Google Colab secret.
```

Defining the Model object

We need to define the Model object using the properties we defined so far:

```
#first, run a quick API test to make sure that endpoint is working
import os
import requests

api_key = os.getenv("IBM_CLOUD_TOKEN")  # or paste your key directly

response = requests.post(
    "https://iam.cloud.ibm.com/identity/token",
    data={
        "grant_type": "urn:ibm:params:oauth:grant-type:apikey",
        "apikey": api_key
    },
    headers={"Content-Type": "application/x-www-form-urlencoded"}
)

print("Status code:", response.status_code)
print("Response body:", response.text[:200])
```

```
Status code: 200
Response body: {"access_token":"eyJraWQiOiIyMDE5MDcyNCIsImFsZyI6IlJTMjU2In0.eyJpYW1faWQiOiJJQk1pZC02NjUwMDBKUVdGIiwiaWQiOiJJQk1p
```

```
from ibm_watsonx_ai.foundation_models import ModelInference

model = ModelInference(
    model_id = model_id,
    params = parameters,
    credentials = credentials,
    project_id = project_id,
    space_id = space_id
    )
```

Defining the vector index

Initialize the vector index to query when chatting with the model.

```
from ibm_watsonx_ai.foundation_models.utils import Toolkit

vector_index_id = "a35ac887-d396-40e5-9d68-bfa3c9d5b45d"

def proximity_search( query ):

    api_client = APIClient(
        project_id=project_id,
        credentials=credentials,
    )

    document_search_tool = Toolkit(
        api_client=api_client
    ).get_tool("RAGQuery")

    config = {
    "vectorIndexId": vector_index_id,
    "projectId": project_id
    }

    results = document_search_tool.run(
        input=query,
        config=config
    )
```

```
        return results.get("output")
```

Defining the inferencing input for chat

Foundation models supporting chat accept a system prompt that instructs the model on how to conduct the dialog. They also accept previous questions and answers to give additional context when inferencing. Each model has it's own string format for constructing the input.

Let us provide the input we got from the Prompt Lab and format it for the selected model:

```
chat_messages = [];
```

Execution

Let us now use the defined Model object, pair it with the input, and generate the response to your question:

```
question = input("Question: ")
grounding = proximity_search(question)
chat_messages.append({
    "role": f"system",
    "content": f"""{grounding}

You are Mixtral Chat, an AI language model developed by Mistral AI. You are a cautious assistant. You carefully follow instructi

"""
})
chat_messages.append({"role": "user", "content": question})
generated_response = model.chat(messages=chat_messages)
print(generated_response)
```

```
Question: I need to cut down on unnecessary expenses so I can save more. Based on my monthly budget (in $), can you see anywhere
{'id': 'chatcmpl-94245826ff3e48e5295cee7cb8e568ac---fa668d9a-799a-4468-90c1-0470dc871df6', 'object': 'chat.completion', 'model_i
```

```
output = generated_response['choices'][0]['message']['content']
```

```
from IPython.display import Markdown, display

def print_markdown(string):
  display(Markdown(string))

print_markdown(output)
```

Based on your monthly budget, here are some suggestions to help you trim down expenses:

1. **Alcohol & Bars ($50)**: Consider reducing the amount spent on alcohol and bars. This is a discretionary expense that can be easily cut down or eliminated.
2. **Coffee Shops ($15)**: Brewing coffee at home can save you a significant amount over time. This is another discretionary expense that can be reduced.
3. **Entertainment ($25)**: Look for free or low-cost entertainment options. This could include community events, outdoor activities, or using free resources like libraries.
4. **Fast Food ($15)**: Reducing fast food expenses can also contribute to savings. Preparing meals at home is generally more cost-effective.
5. **Home Improvement ($250)**: This is a significant expense. If possible, prioritize necessary improvements and postpone non-essential projects.
6. **Music ($11)**: Consider using free music streaming services or reducing subscriptions if you have multiple ones.
7. **Restaurants ($150)**: Similar to fast food, reducing the frequency of dining out can save money. Cooking at home is usually cheaper.
8. **Shopping ($100)**: Evaluate your shopping habits. Look for sales, use coupons, and consider if purchases are truly necessary.
9. **Television ($15)**: If you have multiple streaming services, consider reducing the number of subscriptions.

Final Answer:

By cutting down on discretionary expenses like Alcohol & Bars, Coffee Shops, Entertainment, Fast Food, and Restaurants, you can save a significant amount each month. Additionally, reviewing and potentially reducing expenses in categories like Home Improvement, Music, Shopping, and Television can further increase your savings.

Next steps

You successfully completed this notebook! You learned how to use watsonx.ai inferencing SDK to generate response from the foundation model based on the provided input, model id and model parameters. Check out the official watsonx.ai site for more samples, tutorials, documentation, how-tos, and blog posts.

Copyrights

↳ 1 cell hidden