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

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

```
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
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
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<0.29,>=0.27->ibm_watsonx_a
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->ibr
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<2.1
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_watsor
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas<2.3.0,>=0.24.2->ibm_watson
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas<2.3.0,>=0.24.2->ibm_watsc
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->ibm_watsonx_
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_wat
Requirement already satisfied: typing_extensions>=4.5 in /usr/local/lib/python3.12/dist-packages (from anyio->httpx<0.29,>=0.2
Collecting git+https://github.com/ibm-granite-community/utils
  Cloning https://github.com/ibm-granite-community/utils to /tmp/pip-req-build-k9zxne_1
  Running command git clone --filter=blob:none --quiet https://github.com/ibm-granite-community/utils /tmp/pip-req-build-k9zxn
  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
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==
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-g
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-g
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-granit
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-g
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.12/dist-packages (from langchain_core->ibm-gra
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/dist-packages (from jsonpatch<2.0.0,>=1.33.0->lang
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->lang
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->langch
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->langc
```

```
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.0.0->lang
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.0.0->langsmith<
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packages (from anyio->httpx<1,>=0.23.0->langsmit
Building wheels for collected packages: ibm-granite-community-utils
  Building wheel for ibm-granite-community-utils (pyproject.toml) ... done
  Created wheel for ibm-granite-community-utils: filename=ibm_granite_community_utils-0.1.dev112-py3-none-any.whl size=16277 s
  Stored in directory: /tmp/pip-ephem-wheel-cache-l_ehd7rm/wheels/e2/74/0e/e7dc80cad1c61a0c57be9aff96c6c6bbb058052bc5b9cac0ff
Successfully built ibm-granite-community-utils
Installing collected packages: ibm-granite-community-utils
Successfully installed ibm-granite-community-utils-0.1.dev112
```

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.

```
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:

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

Defining the model parameters

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

```
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:

```
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 = "e0cb43c3-093c-4481-8c97-cc3b0d6c076a"

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 instruct

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

```
Question: Based on my transactional data, can you give me any recommendations on how to better budget and lower my expenses? I w
{'id': 'chatcmpl-6a3844d4504cba2f1f63415cf21194ec---df8d809b-5c61-42df-8025-83d88dd64b93', '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 transactional data, here are some recommendations to help you better budget and lower your expenses:

1. **Identify High-Expense Categories**:
   - **Rent**: This is a significant expense. Consider options like finding a more affordable place to live, getting a roommate, or negotiating your rent.
   - **Food & Drink**: This category also shows high expenses. Plan your meals, cook at home, and reduce eating out.
   - **Health & Fitness**: While important, look for more cost-effective alternatives like home workouts or community fitness programs.
   - **Travel**: This is another area with substantial spending. Plan trips in advance, look for deals, and consider more budget-friendly travel options.
   - **Entertainment**: This category can often be reduced by finding free or low-cost activities, such as community events or outdoor activities.
2. **Track and Categorize Expenses**:
   - Use budgeting tools or apps to track your expenses. Categorize them to see where your money is going each month.
3. **Set Budget Limits**:
   - Based on your income and necessary expenses, set monthly limits for each category. Stick to these limits to avoid overspending.
4. **Reduce Discretionary Spending**:
   - Look at categories like Shopping, Entertainment, and Travel. These are areas where you can often cut back without significantly impacting your quality of life.
5. **Increase Income**:
   - While not directly related to expenses, increasing your income through side jobs, investments, or other means can help you save more.
6. **Review and Adjust Regularly**:
   - Regularly review your budget and expenses. Adjust your spending habits as needed to stay on track with your savings goals.
7. **Emergency Fund**:
   - Ensure you have an emergency fund to cover unexpected expenses. This can help you avoid dipping into your savings or going into debt.

By following these recommendations, you can better manage your expenses and 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