

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

```
Downloading ibm_watsonx_ai-1.4.5-py3-none-any.whl (1.1 MB)
```

```
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4-
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.7.4->
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,>=0.3.4->
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.4->
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.4->
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages (from httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.4->
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.4->
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests>=2.0.0->langsmith<1.0.0,>=0.3.4->
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests>=2.0.0->langsmith<1.0.0,>=0.3.4->
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-packages (from anyio->httpx<1,>=0.23.0->langsmith<1.0.0,>=0.3.4->
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 sha256=
  Stored in directory: /tmp/pip-ephem-wheel-cache-tw8c3nv/wheels/e2/74/0e/e7dc80cad1c61a0c57be9aff96c6c6bbb058052bc5b9cac0fff
Successfully built ibm-granite-community-utils
Installing collected packages: ibm-granite-community-utils
Successfully installed ibm-granite-community-utils-0.1.dev112
```

```
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 = "ibm/granite-3-3-8b-instruct"
```

▼ 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": 0.7,
    "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 = "3506ea1f-b9f9-4dca-a43d-ddd8c073ec57"

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:

```
tone = ""informative""

prompt_input = f""""

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": "user", "content": f""Use the following pieces of context to answer the question.

{grounding}

Question: {question}""})
generated_response = model.chat(messages=chat_messages)
print(generated_response)
```

```
Question: How can I better budget my finances while in college and working part-time? I've been finding it difficult to save and
{'id': 'chatcmp1-144f20d776ca7a539a3a86f5fc79254d---ed14b604-9ebd-4f22-994f-a476bbea827d', '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)
```

To better budget your finances while in college and working part-time, consider the following steps:

1. **Track your income and expenses:** Keep a record of all your income sources (part-time job, refereeing, etc.) and all your expenses (tuition, textbooks, rent, food, entertainment, etc.). Use budgeting apps or spreadsheets to make this process easier.
2. **Categorize your expenses:** Organize your expenses into categories such as 'fixed' (rent, tuition), 'variable' (food, entertainment), and 'one-time' (books, electronics). This will help you identify areas where you might be overspending.
3. **Create a budget plan:** Based on your income and expense tracking, allocate funds for each category in your budget. Allocate a certain percentage for savings, aiming for at least 10-20% of your income. Prioritize essential expenses like rent, tuition, and groceries, and cut back on discretionary spending.
4. **Prioritize paying off student loans:** Make a plan to pay off your student loans. Look into loan consolidation or refinancing options that could potentially lower your monthly payments and interest rates. Make sure to make at least the minimum payments on time to avoid damaging your credit.
5. **Increase your income (if possible):** Explore opportunities to increase your income, such as taking on more hours at your part-time job, finding additional work-study opportunities on campus, or looking for a better-paying position.
6. **Create an emergency fund:** Aim to save at least 500–1,000 as an emergency fund to cover unexpected expenses, such as car repairs or medical bills. Having this buffer will provide peace of mind and prevent you from falling back on credit cards or loans in times of need.
7. **Review and adjust your budget regularly:** Periodically review your budget to ensure it remains realistic and effective. Make adjustments as needed based on changes in your income, expenses, or priorities.
8. **Explore additional financial aid resources:** Investigate other financial aid options such as scholarships, grants, or work-study programs. Many organizations and institutions offer financial assistance specifically for college students.
9. **Seek financial advice:** Consult with a financial advisor, especially if you're struggling to manage your finances and student loan debt. They can provide personalized guidance and strategies tailored to your situation.
10. **Educate yourself on personal finance:** Learn about personal finance concepts such as compound interest, the power of saving early, and the benefits of investing. This knowledge will empower you to make informed decisions regarding your money and set you up for long-term financial success.

By implementing these steps, you'll be better equipped to manage your finances while in college and working part-time, making progress towards paying off student loan debt and achieving your financial goals.

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

Licensed Materials – Copyright © 2023 IBM. This notebook and its source code are released under the terms of the ILAN License. Use, duplication disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Note: The auto-generated notebooks are subject to the International License Agreement for Non-Warranted Programs (or equivalent) and License Information document for watsonx.ai Auto-generated Notebook (License Terms), such agreements located in the link below. Specifically, the Source Components and Sample Materials clause included in the License Information document for watsonx.ai Studio Auto-generated Notebook applies to the auto-generated notebooks.

By downloading, copying, accessing, or otherwise using the materials, you agree to the [License Terms](#)

Start coding or [generate](#) with AI.

