

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

vatsonx 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 lomond-0.3.3-py3-none-any.whl (55 kB)
Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Building wheels for collected packages: ibm-cos-sdk, ibm-cos-sdk-core, ibm-cos
  Building wheel for ibm-cos-sdk (setup.py) ... done
    Created wheel for ibm-cos-sdk: filename=ibm_cos_sdk-2.14.3-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/cc/2f/6f/125918ad46d280d3bea58e
  Building wheel for ibm-cos-sdk-core (setup.py) ... done
    Created wheel for ibm-cos-sdk-core: filename=ibm_cos_sdk_core-2.14.3-py3-none-any.whl
  Stored in directory: /root/.cache/pip/wheels/f1/53/13/7c8fdeebdb847995d8ef34
  Building wheel for ibm-cos-sdk-s3transfer (setup.py) ... done
    Created wheel for ibm-cos-sdk-s3transfer: filename=ibm_cos_sdk_s3transfer-2.
      Stored in directory: /root/.cache/pip/wheels/0c/8b/10/0346c5a955b48b7fcfa50a8
Successfully built ibm-cos-sdk ibm-cos-sdk-core ibm-cos-sdk-s3transfer
Installing collected packages: lomond, jmespath, ibm-cos-sdk-core, ibm-cos-sdk
Successfully installed ibm-cos-sdk-2.14.3 ibm-cos-sdk-core-2.14.3 ibm-cos-sdk-
Collecting git+https://github.com/ibm-granite-community/utils
  Cloning https://github.com/ibm-granite-community/utils to /tmp/pip-req-build
    Running command git clone --filter=blob:none --quiet https://github.com/ibm-granite-community/utils
  Resolved https://github.com/ibm-granite-community/utils to commit 3c725e2435
  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-
Requirement already satisfied: langchain_core in /usr/local/lib/python3.12/dist-
Requirement already satisfied: typing_extensions in /usr/local/lib/python3.12/
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10.0.0,>=8.1.0 in /usr/local/l
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pyth
Requirement already satisfied: PyYAML<7.0.0,>=5.3.0 in /usr/local/lib/python3.
Requirement already satisfied: packaging<26.0.0,>=23.2.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/d
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dis
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/pyth
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.12/di
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/pyth
Requirement already satisfied: anyio in /usr/local/lib/python3.12/dist-package
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-
Requirement already satisfied: idna in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyth
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.12/dist-
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_
      Stored in directory: /tmp/pip-ephem-wheel-cache-u0rwjuvs/wheels/e2/74/0e/e7d
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 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,
    "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 = "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 its 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": "user", "content": f"""Use the following pieces of information to answer the question:  
{grounding}  
  
Question: {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 current spending habits, what are some ways I can reduce my expenses?  
{'id': 'chatcmpl-01277d063a7baa900a8800ff00cce43b---1f0a6ca8-bb40-4fc3-b280-0a70e0c'}  
Answer: There are several ways you can cut down on unnecessary expenses. One way is to track your spending habits and identify areas where you can cut back. You can also consider switching to more affordable alternatives, such as generic brands or public transportation. Another way is to create a budget and stick to it. This will help you stay within your means and avoid overspending. Additionally, you can look for discounts and promotions on items you need. Finally, consider reducing non-essential expenses, such as eating out or buying new clothes, to free up more money for savings.
```

```
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 the provided budget, there are several categories where you could potentially cut down on expenses:

1. Electronics & Software: Your budget for this category is \$0. If you do purchase electronics or software, you could look for deals or consider used items to save money.
2. Movies & DVDs: Your budget here is also \$0. You might consider streaming services that offer more cost-effective plans or free trials.
3. Entertainment: With a budget of \$25, you could explore free or low-cost entertainment options like local parks, free community events, or borrowing books and movies from the library.
4. Fast Food: Spending \$15 monthly on fast food could be reduced by preparing meals at home. This could lead to healthier eating habits and significant savings.
5. Coffee Shops: If you're spending \$15 monthly at coffee shops, making coffee at home or limiting your visits could help cut costs.
6. Restaurants: With a \$150 budget, you might consider cooking at home more often or dining out less frequently.
7. Shopping: If non-grocery shopping falls under this \$100 category, you could try to limit

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](#)