

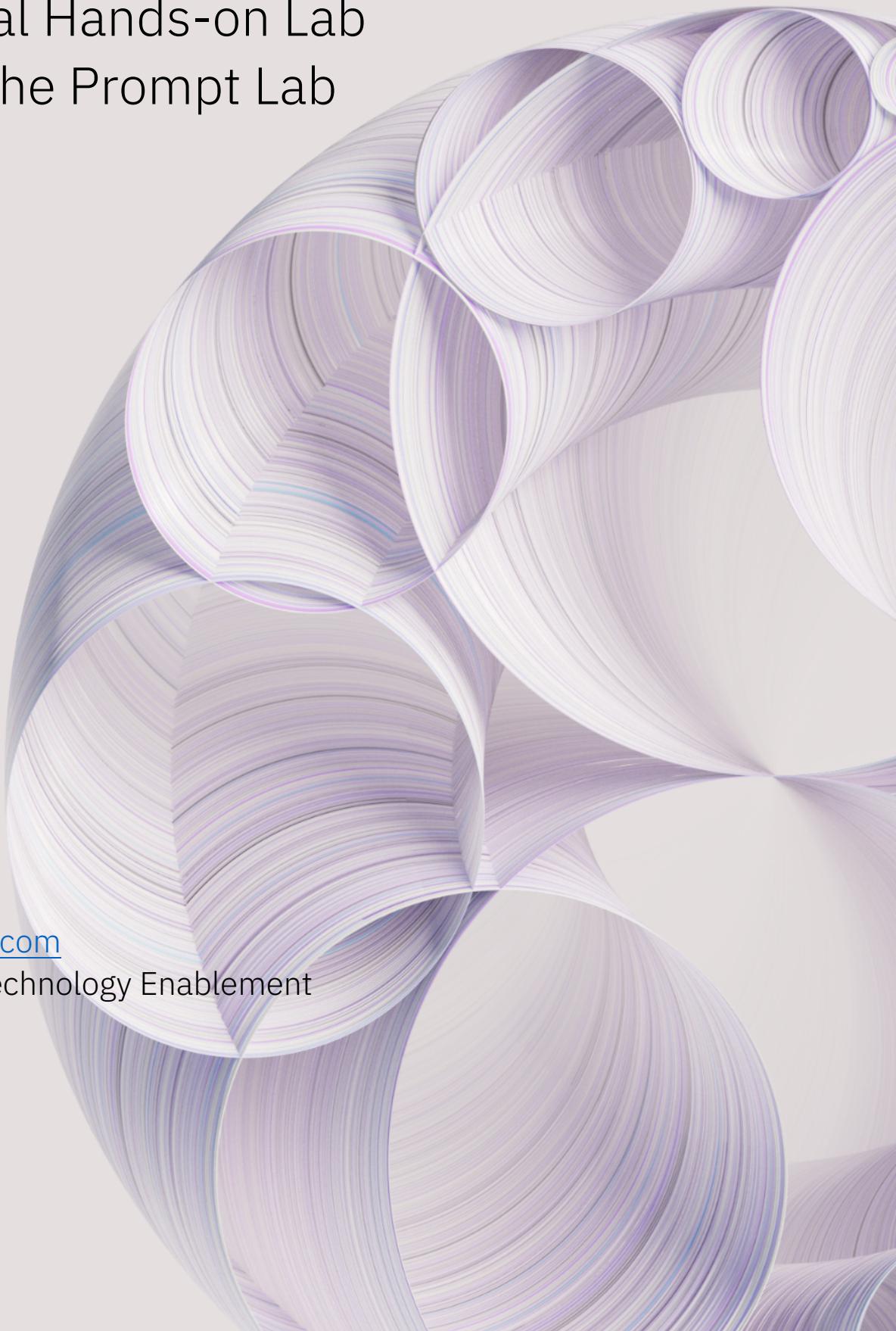
IBM watsonx.ai

Technical Hands-on Lab Chat in the Prompt Lab

Felix Lee

felix@ca.ibm.com

Worldwide Technology Enablement



Contents

1. Introducing watsonx.ai	3
2. About this Lab.....	5
2.1 Disclaimer	5
3. Getting Help.....	6
4. Prerequisites & Getting Started	6
4.1 Obtain an IBM Cloud Account.....	6
4.2 Verify your TechZone account.....	7
5. Chat feature in the Prompt Lab	10
5.1 Accessing the Chat feature on the watsonx.ai console.....	10
5.2 Accessing Chat in the Prompt Lab	13
5.3 Chat session example.....	18
5.4 Different LLMs supported in watsonx.ai Chat	29
5.5 Chat and Retrieval Augmented Generation (RAG)	38
5.5.1 Chat in a RAG use case – inline method	38
5.5.2 Chat with Documents	41
5.5.3 Exploring options for Chat with Documents	49
5.6 A Closer Look at watsonx.ai Internals: memory usage and guardrails	53
5.6.1 Memory handling	53
5.6.2 IBM watsonx.ai memory implementation.....	57
5.6.3 AI guardrails in a Chat	62
Appendix A. Revision History.....	70

1. Introducing watsonx.ai

Watsonx.ai is a core component of Watsonx, IBM's enterprise-ready AI and data platform that's designed to multiply the impact of AI across an enterprise.

The Watsonx platform has three powerful components:

- **watsonx.ai**, a studio for new foundation models, generative AI, and Machine Learning (traditional AI)
- **watsonx.data**, a fit-for-purpose data store that provides the flexibility of a data lake with the performance of a data warehouse
- **watsonx.governance**, a toolkit, which enables AI workflows that are built with responsibility, transparency, and explainability.

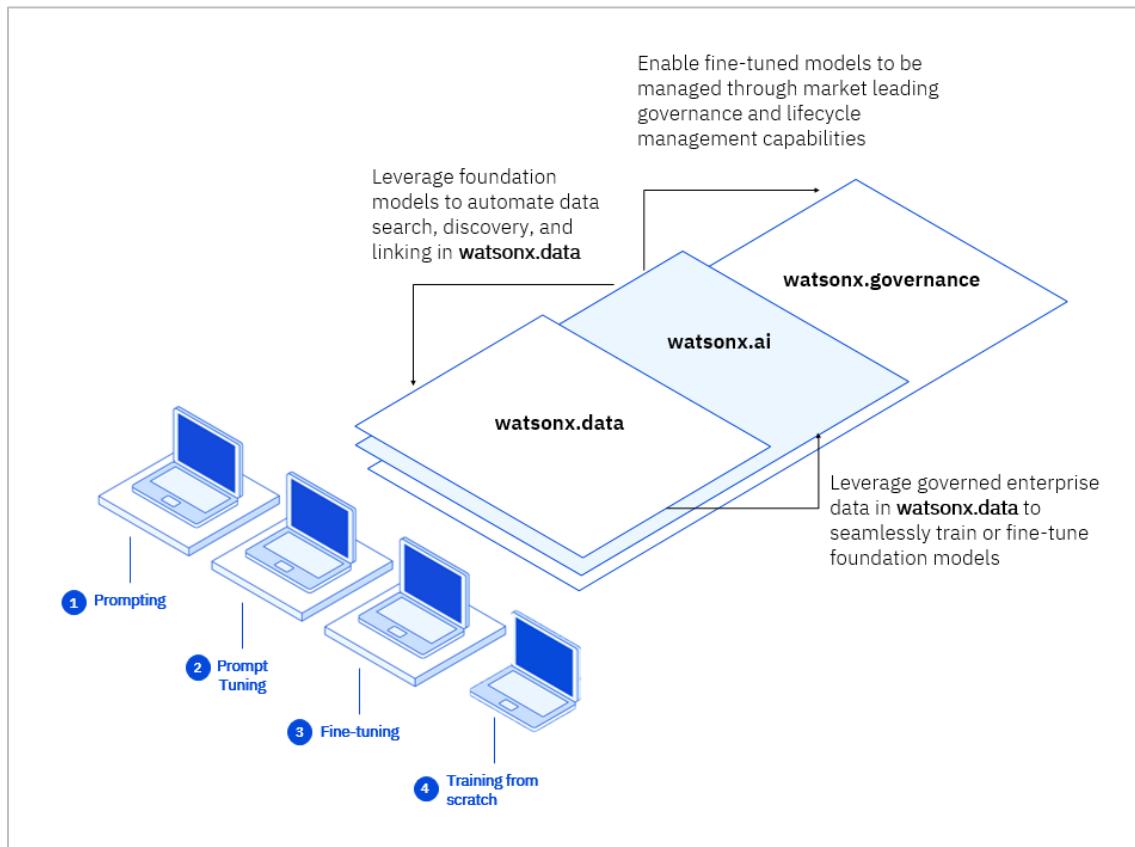


Figure 1: IBM Watsonx platform

The Watsonx.ai component (the focus of this lab) makes it possible for enterprises to train, validate, tune, and deploy AI models – both traditional AI and generative AI. With Watsonx.ai, enterprises can leverage their existing traditional AI investments as well as exploit the innovations and the potential of generative AI using foundation models to bring advanced

automation and AI-infused applications to reduce cost, improve efficiency, scale, and accelerate the impact of AI across their organizations.

2. About this Lab

This IBM watsonx.ai Technical Hands-on Lab – Chat in the Prompt Lab is a supplement to the IBM watsonx.ai Generative AI Tools for Technical Sales L3 Lab. It is assumed that you have completed this [Level 3 Hands-on lab](#). Detailed instructions and screen flow available there are not repeated here.

In this Lab: you will examine watsonx.ai's **Chat** capability. This **Chat** function provides:

- An easy entry point into generative AI
- Great examples of getting generative AI to provide a structured output
- An easy way to experience Retrieval Augmented Generation

Use the same procedure from the [Level 3 Hands-on lab](#) to set up the environment for this lab.

2.1 Disclaimer

IBM watsonx.ai is developed and released in an agile manner. In addition to constantly adding new capabilities, the web interface is likely to change over time. **Therefore, the screenshots used in this lab may not always look exactly like what you see.** You can expect to encounter some of the following:

- Additional foundation models in the library list
- Additional foundation models available for prompt tuning
- Changes in the user interface (location of buttons, text for various fields)
- Additional tabs/buttons

These updates should not affect how the lab works, but have patience and be prepared to explore a little bit until this lab gets updated.

There are three changes, however, that can affect the results. For example:

- IBM is constantly evaluating the models in watsonx.ai, and new models may be added, and others deprecated or removed.
- There is ongoing tuning of the models. If the models themselves are updated, then **some of the results may vary.**
- Foundation models can be very sensitive to input. If you enter slightly different text than what the exercise is using (even if it is just one single word or a modified set of labeled data), the outcome can be very different.

Please post any questions on the [#data-ai-demo-feedback](#) Slack channel (IBMer only). IBM partners can request help at the [Partner Plus Support](#) website.

3. Getting Help

Lab guide help: If you require assistance in interpreting any of the steps in this lab, please post your questions to the [#data-ai-demo-feedback](#) Slack channel (IBMer only). Business Partners can request help at the [Partner Plus Support](#) website.

Troubleshooting: See the [TechZone Set Up Troubleshooting Guide](#) for a list of the common issues and solutions/workarounds when using TechZone.

IBM watsonx.ai: Assistance with the watsonx.ai product itself is available in the [#watsonx-ai-feedback](#) (IBMer) and [#watsonx-ai-enablement](#) Slack channels (IBMer only). Additionally, please refer to the [watsonx.ai documentation](#) as needed.

4. Prerequisites & Getting Started

You will need an IBM Cloud account to gain access to the TechZone account that hosts the various Watson and watsonx services used in this lab.

4.1 Obtain an IBM Cloud Account

If you have an IBM Cloud account, you can skip this step. If you do not have an IBM Cloud account, [Click this link](#) to create one. After registration, you will be sent an email to activate your account. This can take a few hours to process. Once you receive the confirmation email, follow the instructions provided in the email to activate your account.

You will use **IBM TechZone** to perform the exercises in this lab. The detailed setup instructions are available in Section 4 of this [Level 3 Hands-On Lab](#). You must complete the steps before you can proceed with any of the exercises in this lab.

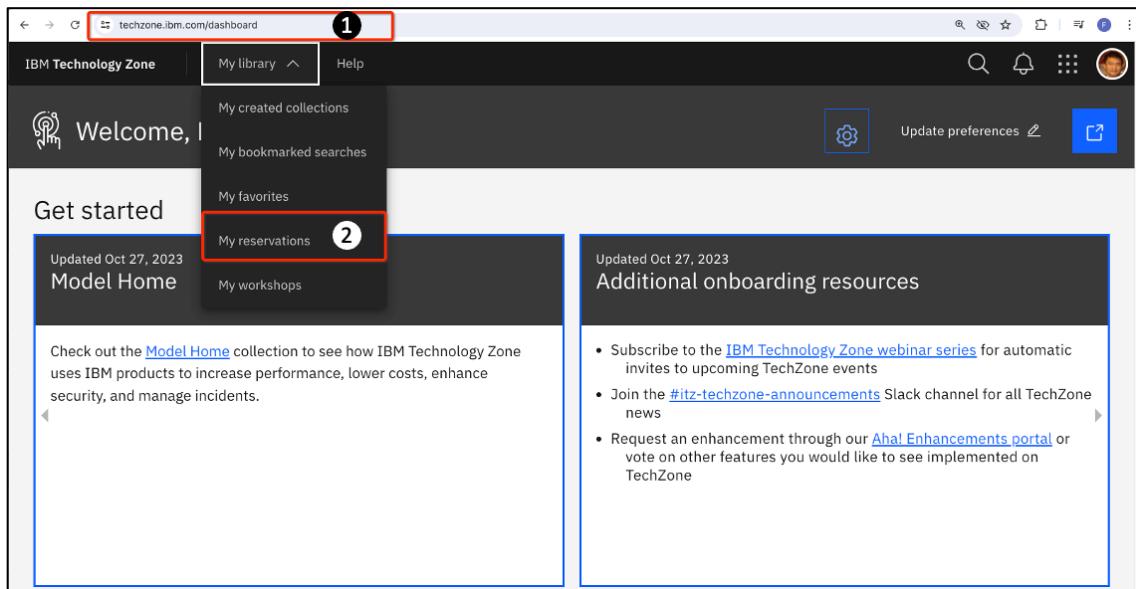
It is assumed that you have completed this [Level 3 Hands-On Lab](#) (or are familiar with the concepts and contents). In particular, you need to be able to:

- Create a sandbox project
- Navigate around the Prompt Lab
- Change to use different models
- Change inference parameters for a model
- Upload files to watsonx.ai projects

4.2 Verify your TechZone account

If you have completed a wastonx.ai L3 lab already, you have already been added to some itz-watsonx-n (n is an integer) account previously. Note that once your old reservation is expired, you cannot use the account anymore. When you make a new reservation, TechZone may assign a different itz-watsonx-n account. It is very important to verify that you are using the proper account for this lab to work.

1. Go to <http://techzone.ibm.com/dashboard>
2. Click **My library**, then click **My reservations**.



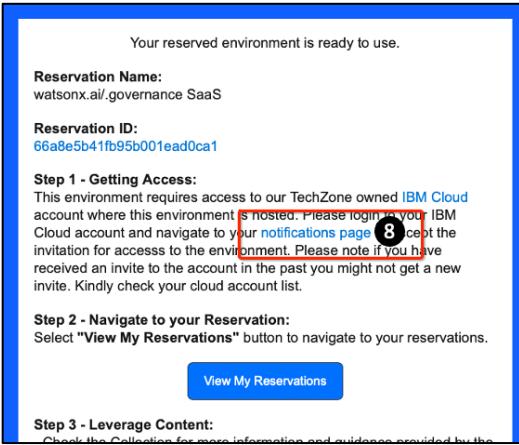
3. The **My reservations** page opens. Look for the **watsonx.ai/.governance SaaS** tile
4. Check the **Status**. If it says **Provisioning**, simply wait a couple of minutes and refresh the page. Repeat the refresh until **Status** is **Ready**.

The screenshot shows the 'My reservations' interface. There are two main sections. The left section displays a reservation for 'Education' with a status of 'Ready'. The right section displays a reservation for 'Education' with a status of 'Ready'. Both sections include fields for 'Username' (e.g., itzuser), 'Password' (e.g., USE SSH KEY), and a 'Status: Ready' indicator.

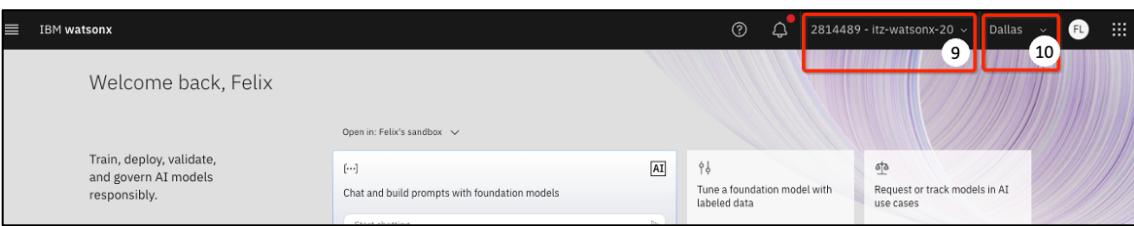
5. Click on the tile once **Status** is **Ready**.
6. The detailed reservation page appears. Check the value of **Cloud Account**. In this example, it is **ITZ-WATSONX-20**.
7. Check the value of **Region**. In this example, it is **us-south**.

The screenshot shows the detailed view of a reservation for 'watsonx.ai/.governance SaaS'. The page includes sections for 'Purpose' (with 'Education' selected) and 'Environment' (with 'Cloud Account' set to 'ITZ-WATSONX-20' and 'Region' set to 'us-south'). Other visible fields include 'Reservation ID', 'Request method', 'Opportunity ID(s)', 'Opportunity description', 'Type', 'Transaction ID', 'Geo', 'Datacenter', and 'Environment'.

8. Check your email and look for an invitation to join the account **ITZ-WATSONX-20**. You might not get one if TechZone has previously assigned it to you in another reservation. If you do receive an invitation to join the account, click **notifications page** and follow the steps to join the account. If you have done so already, skip to Step 9.



9. When you log into watsonx.ai, ensure you are using the right **Account**: in this example it is **ITZ-WATSONX-20**.
10. Verify that you are in the right **Region**: in this case **us-south** (or **Dallas**).



If either the **Account** or **Region** is wrong, this lab would not work.

11. Note that the **Region** value from the **TechZone** reservation page is different from what is shown on watsonx.ai. Use the following table for the mapping value:

Value from TechZone detail page	Value on watsonx.ai User Interface
us-south	Dallas
eu-de	Frankfurt
eu-gb	London
jp-tok	Tokyo

5. Chat feature in the Prompt Lab

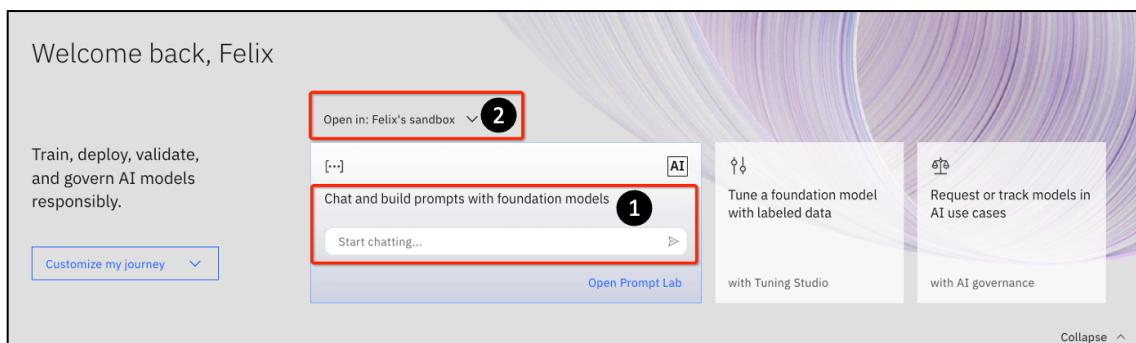
There are different ways to interact with watsonx.ai large language models (LLMs). In the Generative AI Tools lab, you worked with the **Structured** and **Freeform** interfaces. The third way to interact with watsonx.ai's Large Language Models (LLMs) is to use its **Chat** user interface (UI) from the Prompt Lab.

In the **Chat**, a user carries out a chat interaction with an LLM (typically, but not exclusively, a chat model). **Chat** is designed to enhance developer experience with LLMs, especially when they are developing applications that involve real-time user interactions. Developers would be able to try out different models rapidly.

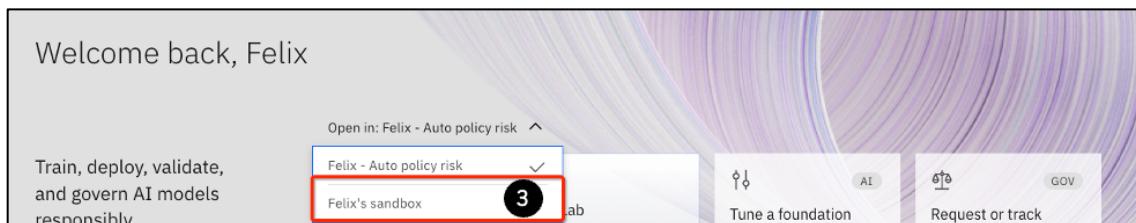
There are various examples you can try out in the **Chat**. You can take the code snippet from these examples and customize it with different models and settings for your use case.

5.1 Accessing the Chat feature on the watsonx.ai console

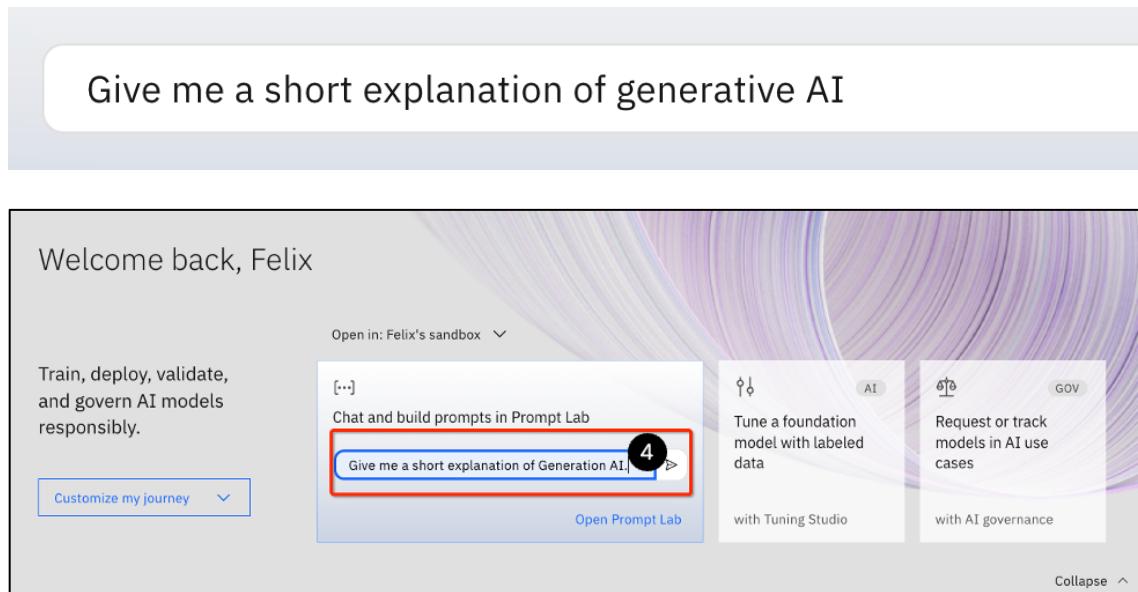
1. The **Chat** feature is accessible immediately when you log onto the watsonx.ai console.
2. The **Chat** function is run in the context of a project using resources such as tokens and Capacity Unit Hours (CUH) from that project. If you have more than one project, watsonx.ai will by default use the first project on the list. You can select what project to use by opening up the drop-down menu and choosing the appropriate project.



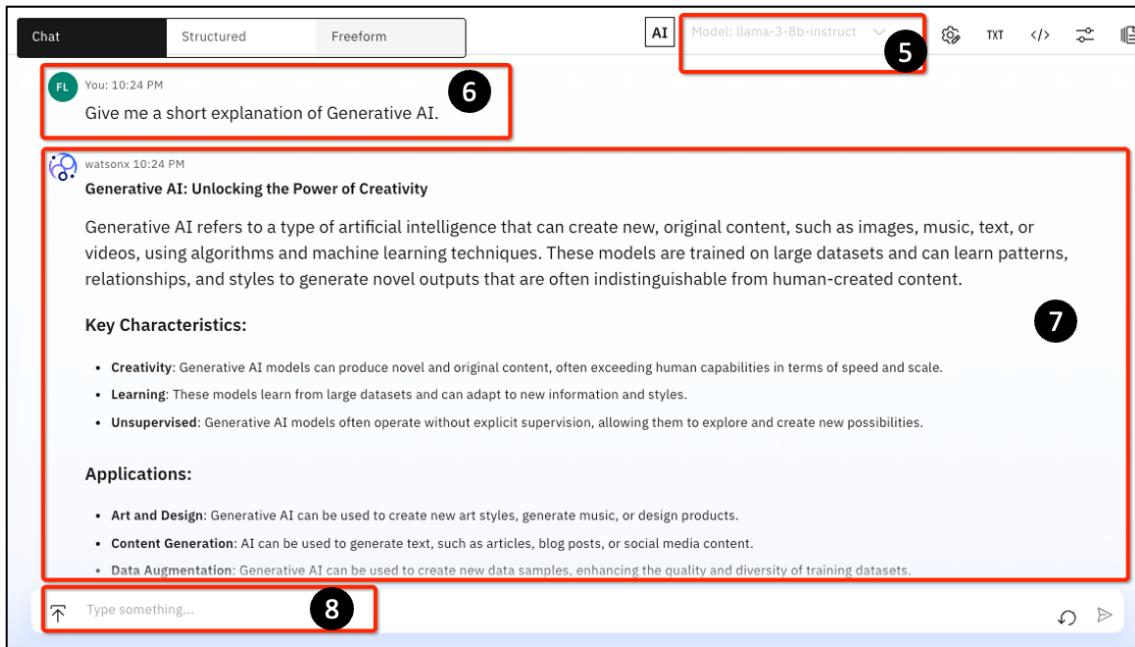
3. When using a TechZone account, you should have a sandbox project. If you are not using the sandbox project, click on the drop-down and select the sandbox project. If it is already listing the sandbox project, proceed to Step 4.



4. Enter the following text into the **Chat** input field then press **Enter**.



5. The **Chat** page opens. The model being used is **llama-3-8b-instruct**. Notice that the name is greyed out (see screen capture below). Once you have started a chat session, the model as well as the inference parameters are locked and cannot be changed (more on this later).
6. The input prompt text is shown on the page.
7. It may take a couple of seconds to finish the completion (partially shown below). The **Max tokens** value is set high in a chat session, and the volume of text being generated can be quite large.
8. You can enter another prompt and have a “chat” with the LLM.



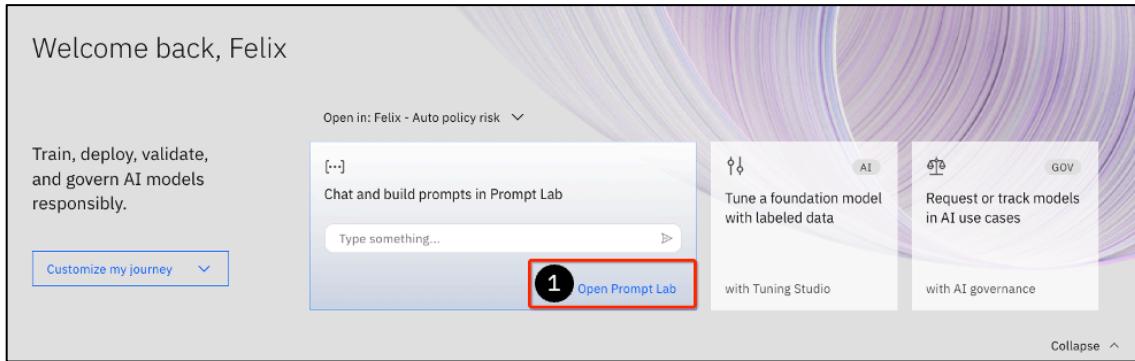
When you start using **Chat** this way, you will not see the examples and you won't be able to choose the model you want to use. At a high level, a Chat is stateful. The LLM needs to remember the context of the conversation, so the model (as well as inference parameter settings) needs to be conserved.

Consider a customer calling a support center. That customer needs to be served by the same agent and not be switched to a different agent every time they say something, as the new agent would not have the context of the previous conversation. In a later section, you will examine the code being used in a chat and the reasoning will become clearer.

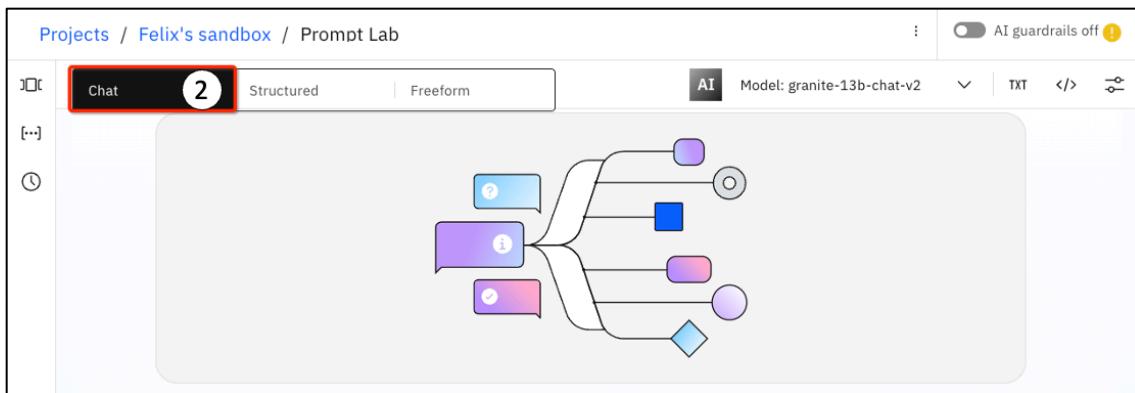
In the next section, you will access **Chat** via the Prompt Lab.

5.2 Accessing Chat in the Prompt Lab

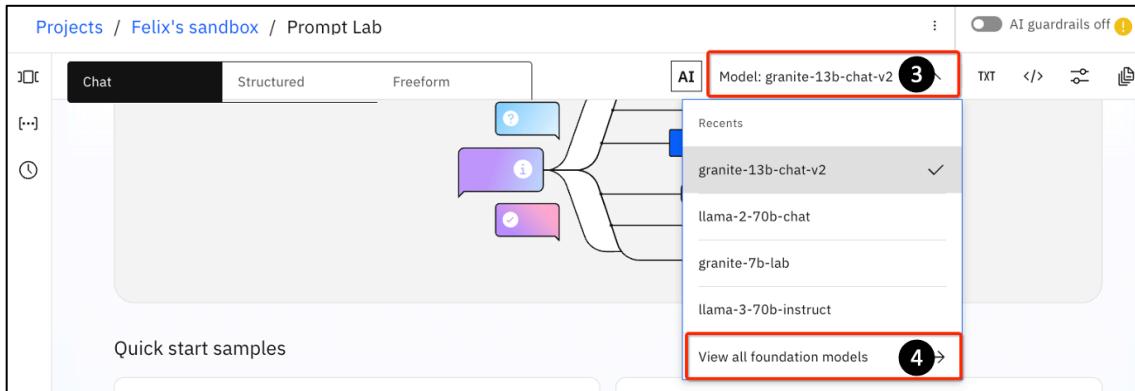
1. Open a new watsonx.ai session and click **Open Prompt Lab** on the first tile.



2. If you are signed into a new watsonx.ai session, watsonx.ai opens in **Chat** mode. If not, you can click on the **Chat** tab.



3. Note that the model shown here is **granite-13b-chat-v2**. You may see a different one; watsonx.ai always show the last model you used.
4. Click on the drop-down list and then select **View all foundation models**.



- Notice that only a subset of all the watsonx.ai LLMs appear in this list. For example, the codellama-34b-instruct-hf and mixtral-8x7b-instruct-v01 models are not here.

Why? Because This is the **Chat** filter is turned on. On the **Select a foundation model** page, this watsonx.ai filter is applied to only show models that are tagged to address the **Chat** task.

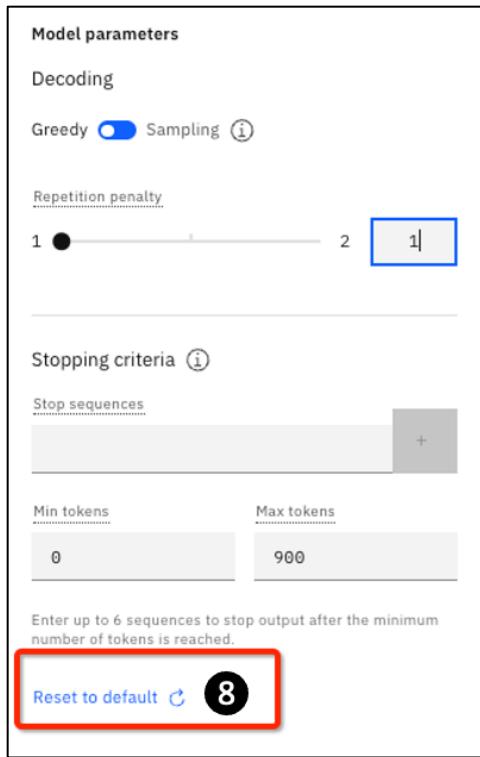
As models will be added and removed from the watsonx.ai list, you might see a different list. This lab starts with **granite-13b-chat-v2**. It can be useful to try different models to see how different models perform for the particular **Chat** task you are working through.

- For now, simply close the **Select a foundation model** page.

Model	Description	Provider	Type
granite-13b-chat-v2	The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative tasks.	IBM	InstructLab
merlinite-7b	Merlinite-7b is a Mistral-7b-derivative model trained with the LAB methodology, using Mixtral-8x7b-Instruct as a teacher model.	Mistral AI, tuned by IBM	InstructLab
granite-13b-instruct-v2	The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative tasks.	IBM	Provided model
granite-20b-code-instruct	The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative tasks.	IBM	Provided model
granite-34b-code-instruct	The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative tasks.	IBM	Provided model
granite-3b-code-instruct	The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative tasks.	IBM	Provided model
granite-8b-code-instruct	The Granite model series is a family of IBM-trained, dense decoder-only models, which are particularly well-suited for generative tasks.	IBM	Provided model
llama-2-13b-chat	Llama-2-13b-chat is an auto-regressive language model that uses an optimized transformer architecture.	Meta	Provided model
llama-2-70b-chat	Llama-2-70b-chat is an auto-regressive language model.		
llama-3-70b-instruct	Llama-3-70b-instruct is an auto-regressive language model.		
llama-3-8b-instruct	Llama-3-8b-instruct is an auto-regressive language model.		
mixtral-8x7b-instruct-v01	The Mixtral-8x7B Large Language Model (LLM) is a...		

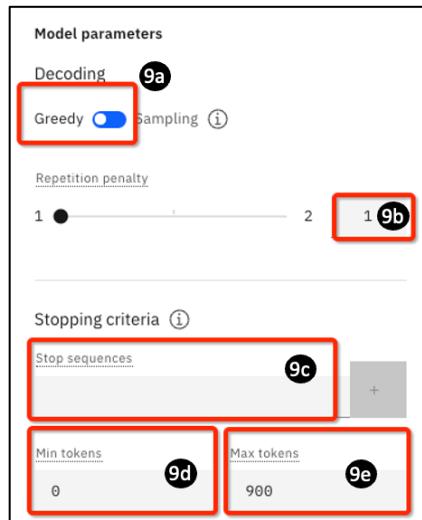
- Click on the **Model parameters** slide-out (the slide-out icon looks like this:).

- Depending on what you might have done previously, you might have different values, click on **Reset to default**. Note that if you have never changed any values, you will not see the **Reset to default** button.



9. The default values are as follows:

- Decoding:** Greedy. Watsonx.ai returns the most probable completion.
- Repetition penalty:** 1. Sometimes a large language model (LLM) may return the same information repeatedly and that can be consuming tokens (and generating costs) needlessly. A setting of 1 allows repetition. You can set it higher if you wish to lower the chance that the model will generate the same text.
- Stop sequences:** none defined
- Min tokens:** 0
- Max tokens:** 900



You can change **Decoding** from **Greedy** to **Sampling**. If you do so you will see additional inferencing parameters that you can modify such as **Temperature**, **Top P**, **Top K**, **Random Seed**, and **Stopping criteria**. These are covered in detail in this [Level 3 Hands-on Lab](#) and will not be discussed further here.

10. Depending on what you did in previous watsonx.ai sessions, you might have turned off the **AI guardrails**. If so, set the AI guardrails switch to on (when guardrails are off, you will see a yellow exclamation mark icon beside the switch).



11. The **Chat** provides a set of samples (the list may change):

These are very useful in finding out how LLMs can provide great output both in content and format. You will try out some of these next.

12. Select the **Describe generative AI using emojis** sample. You will see the following (partially shown) output. Note that when you select one of the 4 samples, watsonx.ai automatically selects the LLM to use. For this example, it is the **llama-3-8b-instruct** model.

The screenshot shows the watsonx.ai interface with the 'Chat' tab selected. At the top, there are three tabs: 'Chat' (selected), 'Structured', and 'Freeform'. On the right, there are buttons for 'AI', 'Model: llama-3-8b-instruct', 'TXT', and other options. A red box highlights the 'Model: llama-3-8b-instruct' button, and a black circle with the number '12' is overlaid on the 'AI' button. The main area shows a conversation between 'You' and 'watsonx'. You sent the message 'Describe generative AI using emojis.' watsonx responded with 'Generative AI 🤖💻'. Below this, there are sections for 'What is it? 🤔' and 'How does it work? 🔬'. The 'What is it?' section defines Generative AI as a type of artificial intelligence that can create new, original content (emojis), such as: Images (camera), Music (speaker), Text (document), and Videos (camera). The 'How does it work?' section explains that Generative AI uses complex algorithms and machine learning techniques to analyze existing data and patterns, and then generates new content based on that analysis (chart). A blue input field at the bottom is labeled 'Type something...'. A red box highlights the 'Model: llama-3-8b-instruct' button, and a black circle with the number '12' is overlaid on the 'AI' button.

13. Clear the chat by clicking on **Clear Chat**.

The screenshot shows the watsonx.ai interface with the 'Chat' tab selected. The main area displays the 'How does it work?' section of the 'Generative AI' sample, which states: 'Generative AI uses complex algorithms and machine learning techniques to analyze existing data and patterns, and then generates new content based on that analysis (chart)'. Below this, there is a blue input field labeled 'Type something...' and a 'Clear Chat' button. A red box highlights the 'Clear Chat' button, and a black circle with the number '13' is overlaid on the button.

14. You will be back to list of examples shown in Step 11.

Section summary

In this section, you looked at how to access **Chat** in watsonx.ai, the LLMs watsonx.ai has tagged for the **Questions answering** use case, and the model inference parameters.

You can experiment with each of the 4 provided Chat examples. After the initial completion, you can carry on a chat with the LLM from there on.

5.3 Chat session example

In this section, you will look at different chat examples provided by watsonx.ai. One of the most useful pieces of information is how to provide instructions for the LLM to output in the format you desire.

5. From the list of chat examples, click on the first one: **What are more efficient alternatives to a ‘for loop’ in Python?**

The screenshot shows a user interface for a chat application. At the top, it says "Quick start samples". Below that, there are four cards, each with a blue arrow pointing to the right. The first card, which is highlighted with a red rounded rectangle, contains the text "What are more efficient alternatives to a 'for loop' in Python?". To its right is a large black circle containing the number "1". The other three cards contain the text "What is the Transformers architecture?", "Create a chart of the top NLP use-cases for foundation models.", and "Describe generative AI using emojis." respectively.

6. You will see the following output (only partially captured here). The completion is in a well-structured format:
 - a. The input prompt is highlighted at the top in a grey box.
 - b. Various alternatives are numbered, making them easier to spot
 - c. Various Python elements are highlighted in grey.
 - d. Code snippets are provided in markdown format

You: 11:10 AM
What are more efficient alternatives to a 'for loop' in Python? **2a**

watsonx 11:10 AM
Efficient Alternatives to a 'for loop' in Python

Python provides several alternatives to traditional **for** loops that can be more efficient in certain situations. Here are a few examples:

1. List Comprehensions **2b**

List comprehensions are a concise way to create lists from existing lists or other iterables. They can be faster than traditional **for** loops especially for large datasets.

```
squares = [x**2 for x in range(10)]
print(squares) # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

2. Generators **2b**

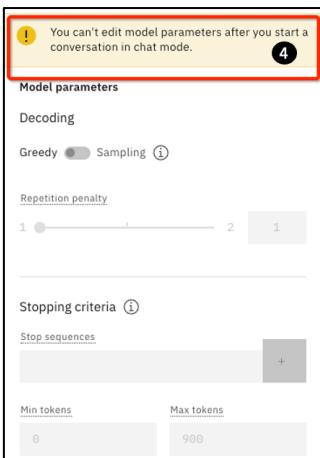
Generators are a type of iterable that can be used to generate a sequence of values on-the-fly. They can be more memory-efficient than traditional **for** loops especially for large datasets.

```
def fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        yield a
```

2c

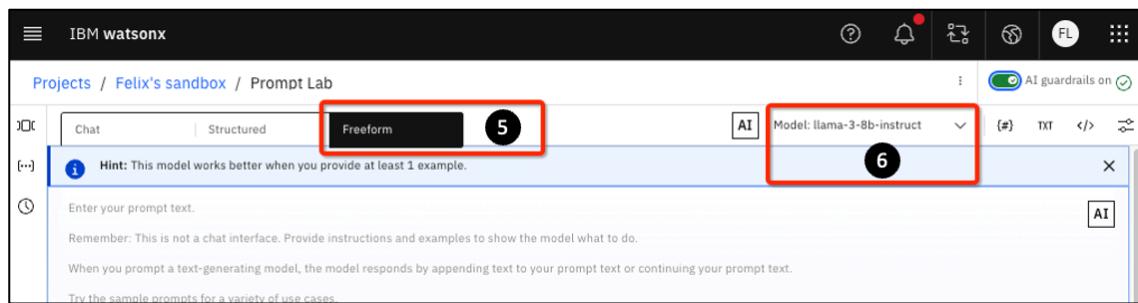
2d

3. The **llama-3-8b-instruct** model was automatically chosen for this sample prompt and it returned 4 alternatives to a Python “for loop”:
 - a. List Comprehensions
 - b. Generators
 - c. Map and Filter Functions
 - d. itertools Functions
4. IBM watsonx.ai may change some of the default inferencing parameter settings at the **beginning** of your chat session. Once you start a conversation in **Chat** mode, you **can no longer change** parameters such as **Decoding** methods, or things like **Max tokens**.

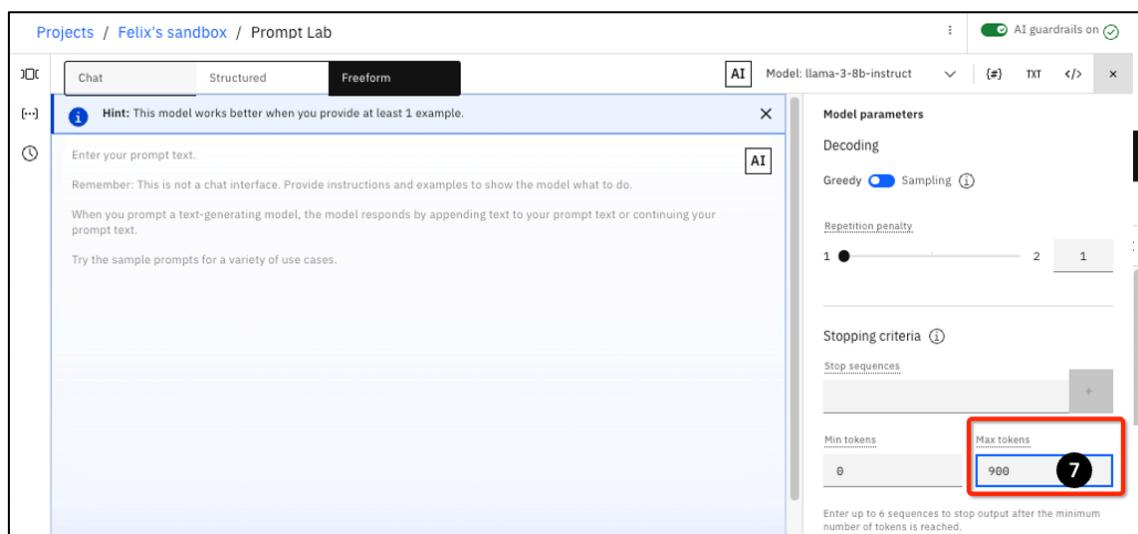


Now you will try the same prompt in the **Structured** mode. Do not clear this generated text or the prompt. You will return to it.

5. From the current **Prompt Lab session** click the **Freeform** tab.
6. Select the **llama-3-8b-instruct** model to use (if not already; this is so you can compare it to the output from **Chat**).



7. Ensure that **Max tokens** is set to 900 in the **Model parameters** (you can access it like you did in Step 7 of Section 5.2).

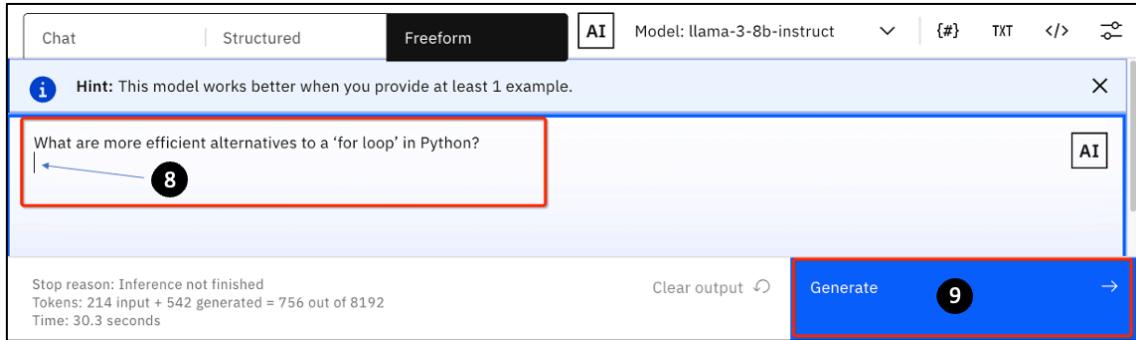


8. Enter the same prompt as the **Chat** example. Be sure to include the carriage return (represented by the extra empty row below what you typed, and by the position of the cursor in the screen capture).

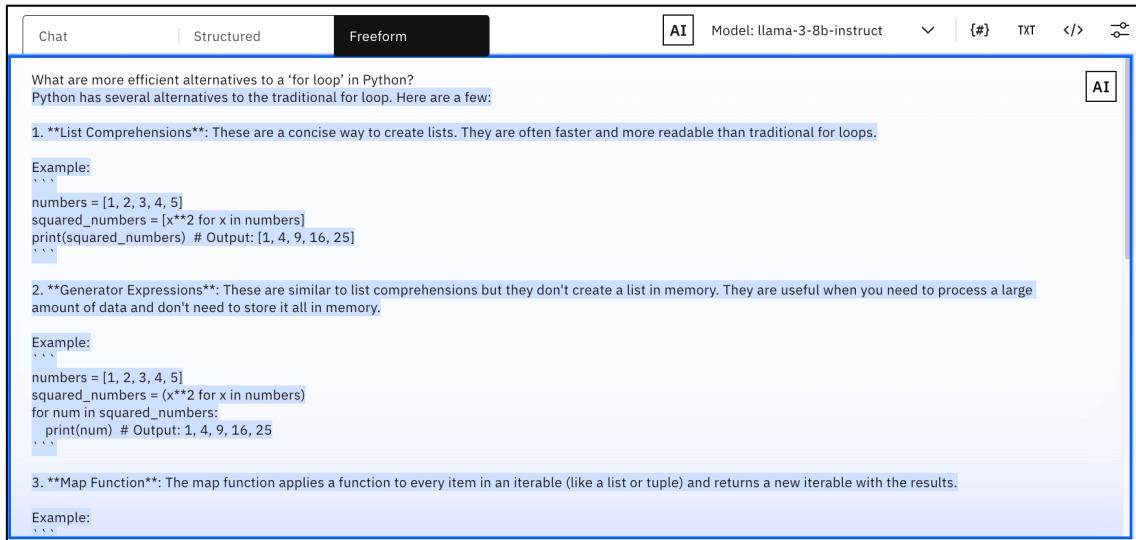
What are more efficient alternatives to a ‘for loop’ in Python?

Bonus exercise: What will the model generate if you do not have the carriage return?

9. Click **Generate**.



You will get this completion (partially shown below):



10. Both the **Structured** and the **Chat** sessions use the **llama-3-8b-instruct** model. There are similarities between the 2 completions: both provide numbered output, and the alternatives and descriptions are similar. However, notice here **Chat** and **Freeform** provide different alternatives for the “for loop”:

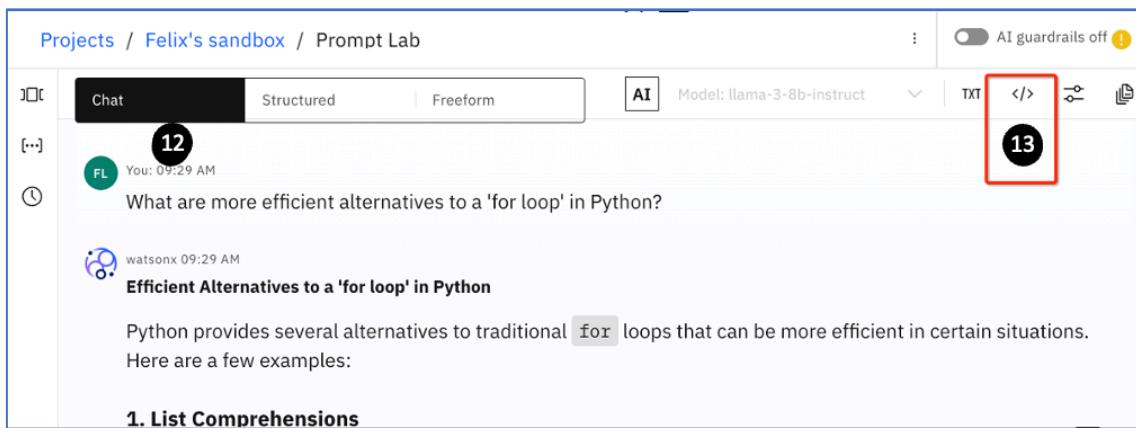
Chat	Freeform
<ul style="list-style-type: none"> • List Comprehensions • Generators • Map and Filter Functions • itertools Functions 	<ul style="list-style-type: none"> • List Comprehensions • Generator Expressions • Map Function • Filter Function • Zip Function • enumerate Function

The **Chat** completion is more structured and easier to read and to extract information from. Also, note how the code examples are in markdown block format.

11. Both **Freeform** and **Chat** used the same model as well as the same inference parameters. The difference is how **Chat** has augmented the prompt to provide a clear, formatted output.

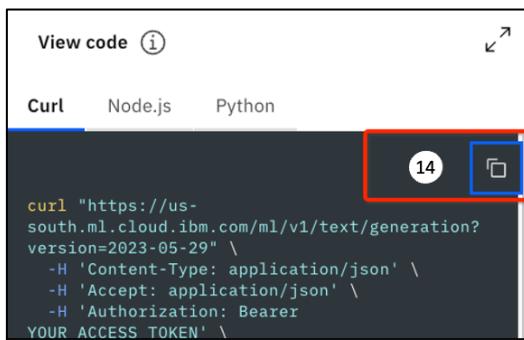
Next, you will add the same augmented text to the **Freeform** prompt session as instructions and observe how it works.

12. Click **Chat** to return to the **Chat** tab. You should still have the sample prompt output there. If not, you can re-run Step 1 to generate the completion again.
13. Click on the **View code** icon (`</>`) icon on the right to show what watsonx.ai is adding to the prompt in the **Chat** mode.



The screenshot shows the WatsonX AI interface. At the top, there's a navigation bar with 'Projects / Felix's sandbox / Prompt Lab'. Below it, a toolbar with tabs: Chat (selected), Structured, Freeform, and AI. To the right of the toolbar are buttons for Model selection (set to 'llama-3-8b-instruct'), AI guardrails (off), and file operations (TXT, copy, etc.). A red box highlights the 'View code' icon (a code tag with </>), which is labeled '13'. In the main workspace, a user message 'You' at 09:29 AM asks: 'What are more efficient alternatives to a 'for loop' in Python?'. An AI response from 'watsonx' at 09:29 AM provides an answer: 'Efficient Alternatives to a 'for loop' in Python'. It includes a link to '1. List Comprehensions'.

14. IBM watsonx.ai shows the underlying code for each LLM interaction. If you are using Python code, building a Jupyter Notebook, or simply running from a command line, you may wish to copy this entire code snippet by clicking on the **copy** icon (`C`), and then pasting the output into a file.



```
View code ⓘ  
Curl Node.js Python  
curl "https://us-south.ml.cloud.ibm.com/ml/v1/text/generation?  
version=2023-05-29" \  
-H 'Content-Type: application/json' \  
-H 'Accept: application/json' \  
-H 'Authorization: Bearer  
YOUR ACCESS TOKEN' \  
14
```

You can modify this snippet as well by adding/updating the instructions.

15. For this lab, Click the **TXT** icon to get a text version of this output.



You can copy the text with the **copy** icon (same one as in Step 15).

16. The text shown below is the set of instructions added by watsonx.ai for the **Chat** session.

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
```

You always answer the questions with markdown formatting using GitHub syntax. The markdown formatting you support: headings, bold, italic, links, tables, lists, code blocks, and blockquotes. You must omit that you answer the questions with markdown.

Any HTML tags must be wrapped in block quotes, for example ` ``<html>`` `. You will be penalized for not rendering code in block quotes.

When returning code blocks, specify language.

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

17. Now go back to the **Freeform** tab.

18. Add this text from Step 16 to the top of the input field. It should look like this:

Chat | Structured | **Freeform** 17

AI Model: llama-3-8b-instruct | {#} **TXT** </> AI

Hint: This model works better when you provide at least 1 example.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You always answer the questions with markdown formatting using GitHub syntax. The markdown formatting you support: headings, bold, italic, links, tables, lists, code blocks, and blockquotes. You must omit that you answer the questions with markdown.

Any HTML tags must be wrapped in block quotes, for example ` ``<html>`` `. You will be penalized for not rendering code in block quotes.

When returning code blocks, specify language.

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

19. Go back to the **Chat** tab and open up the **TXT** slideout again. This time copy the the text associated with this prompt. The text from **Chat** is as follows:

What are more efficient alternatives to a ‘for loop’ in Python?<|eot_id|>

View full prompt text 19

When returning code blocks, specify language

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

What are more efficient alternatives to a 'for loop' in Python?<|eot_id|>

```
<|start_header_id|>assistant<|end_header_id|>
## Efficient Alternatives to a 'for loop' in Python

Python provides several alternatives to traditional 'for' loops that can be more efficient in certain situations. Here are a
```

20. Now go back to the **Freeform** tab. **Enter** two carriage returns to provide a line separation. Then append the text you just copied in Step 19. You should see the screenshot below (with the appended text highlighted with a red box).

21. Click **Generate**.

<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You always answer the questions with markdown formatting using GitHub syntax. The markdown formatting you support: headings, bold, italic, links, tables, lists, code blocks, and blockquotes. You must omit that you answer the questions with markdown.

Any HTML tags must be wrapped in block quotes, for example `'''<html>'''`. You will be penalized for not rendering code in block quotes.

When returning code blocks, specify language.

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

What are more efficient alternatives to a 'for loop' in Python?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Stop reason: End of sequence token encountered
Tokens: 217 input + 607 generated = 824 out of 8192
Time: 8.1 seconds

Clear output ↻ Generate →

20 21

22. Note that your instructions include these points:

- Markdown format is specified, creating a basic framework of output format
- Specific markdown elements are specified, limiting how the LLM can generate completion
- The tone (helpful, respectful, and honest) and manner of completion (safe, not harmful, unethical, etc. in the output) are specified
- Ask for clarification if the input prompt is unclear
- Do not make things up if the LLM does not know how to respond.

This is the generated completion (only partially shown):

Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.<|eot_id|><|start_header_id|>user<|end_header_id|>

What are more efficient alternatives to a 'for loop' in Python?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

Efficient Alternatives to a 'for loop' in Python

Python provides several alternatives to traditional 'for' loops that can be more efficient in certain situations. Here are a few examples:

1. List Comprehensions

List comprehensions are a concise way to create lists from existing lists or other iterables. They can be faster than traditional 'for' loops, especially for large datasets.

```
```python
squares = [x**2 for x in range(10)]
print(squares) # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```
```

2. Generators

Generators are a type of iterable that can be used to generate a sequence of values on-the-fly. They can be more memory-efficient than traditional 'for' loops, especially for large datasets.

```
```python
def fibonacci(n):
 a, b = 0, 1
 for _ in range(n):
 yield a
 a, b = b, a + b
```
```

This time, the **Freeform** mode shows the same 4 options:

- List Comprehensions
- Generators
- Map and Filter Functions
- itertools Functions

You may notice some (very) minor differences but with the same instructions input, you do get the same alternatives and examples in the **Freeform** mode as the **Chat** mode.

23. Now it's time to try different **Chat** samples. First, go back to **Chat**, and click on the **Clear Chat** icon (⟳) at the right-hand side of the input text box to clear all input and return to the list of samples.



24. Click **Create a chart of the top NLP use cases for foundation models**. This is the (partial) output:

 A screenshot of the WatsonX AI interface showing a generated chart titled 'Top NLP Use-Cases for Foundation Models'. The chart is a table with five rows and three columns. The columns are labeled 'Use-Case', 'Description', and 'Applications'. The rows contain the following data:

| Use-Case | Description | Applications |
|--------------------------------|---|--|
| Language Translation | Foundation models can be fine-tuned for machine translation tasks, enabling high-quality translations between languages. | Google Translate, Microsoft Translator, iTranslate |
| Sentiment Analysis | Foundation models can be used for sentiment analysis, identifying the emotional tone of text, such as positive, negative, or neutral. | Customer feedback analysis, social media monitoring, product review analysis |
| Named Entity Recognition (NER) | Foundation models can be trained for NER, identifying and categorizing named entities such as people, organizations, and locations. | Information extraction, text summarization, question answering |
| Question Answering | Foundation models can be fine-tuned for question answering tasks, providing accurate answers to natural language questions. | Virtual assistants, chatbots, search engines |

Note several points:

- This example continues to use the **llama-3-8b-instruct** model. If you check the parameters, it also uses **Max tokens 900**.
- The prompt specified **Create a chart**, and the LLM generated a very neat and clear table with proper headings.

21. It is again useful to look at what code watsonx.ai has added to get the LLM to respond in this manner. Click on the **View code icon** (</>) and look at the code (or copy it). You will see the following:

You always answer the questions with markdown formatting. The markdown formatting you support: headings, bold, italic, links, tables, lists, code blocks, and blockquotes. You must omit that you answer the questions with markdown.

Any HTML tags must be wrapped in block quotes, for example ``<html>``. You will be penalized for not rendering code in block quotes.

When returning code blocks, specify language.

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe.

Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.

Create a chart of the top NLP use-cases for foundation models.

This is the same text from Step 16. If you tried the other 2 sample prompts and look at the code, they would all be identical as well.

The net is that this set of instructions together with the **llama-3-8b-instruct** form a powerful combination for creating structured output using markdown.

Section summary

- The watsonx.ai **Chat** supports the interactive chat use case. Note that watsonx.ai automatically selects a subset of available models for this task. Models such as mixtral-8x7b-instruct-v01-q are filtered out.
- **Chat** shows some fundamental capabilities for LLMs such as **llama-3-8b-instruct** in supporting interactive sessions with clients.
- Using an LLM (such as **llama-3-8b-instruct**) that understands markdown, it is fairly simple to provide a standard set of instructions to augment an input prompt to generate highly formatted output.
- You can copy the instructions from the **Chat** example and use them in the **Freeform** or **Structured** mode to customize it and try different models or model parameters.
- You can configure a chat session before starting a chat which includes the choice of LLM and various inference parameters. Once you started the chat session, you cannot change to a different model or modify the parameters.
- Note that in **Chat** mode, the value of **Max tokens** is set to a high value (can be as high as 900). While this may be desirable in actual production, clients (especially developers) should be aware that this can be costly.
- In this section, you used the **Chat** instructions with the **Freeform** mode. You can also try to copy the instructions into the **Structured** mode. However, it will not work as well because the **Structured** mode includes its own underlying instruction text and structure, and it will interfere with the **Chat** instructions.

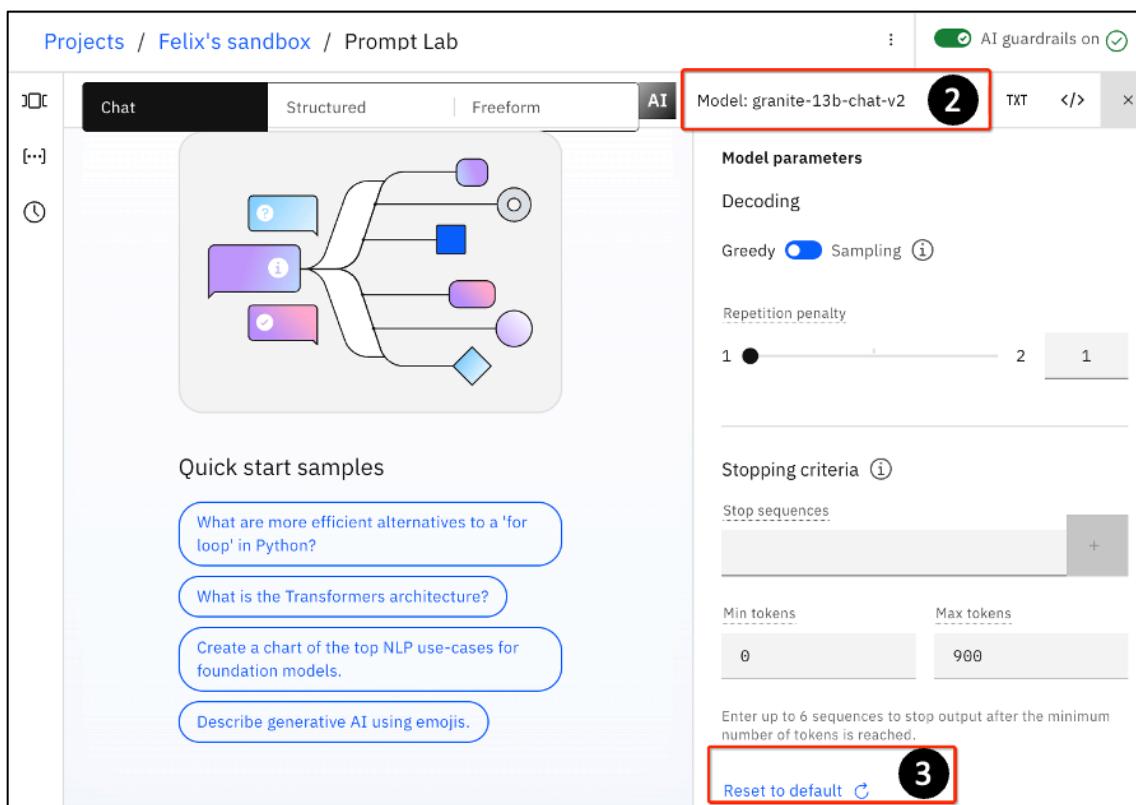
5.4 Different LLMs supported in watsonx.ai Chat

In this section, you will look at different models and how watsonx.ai provides additional instructions to the LLM.

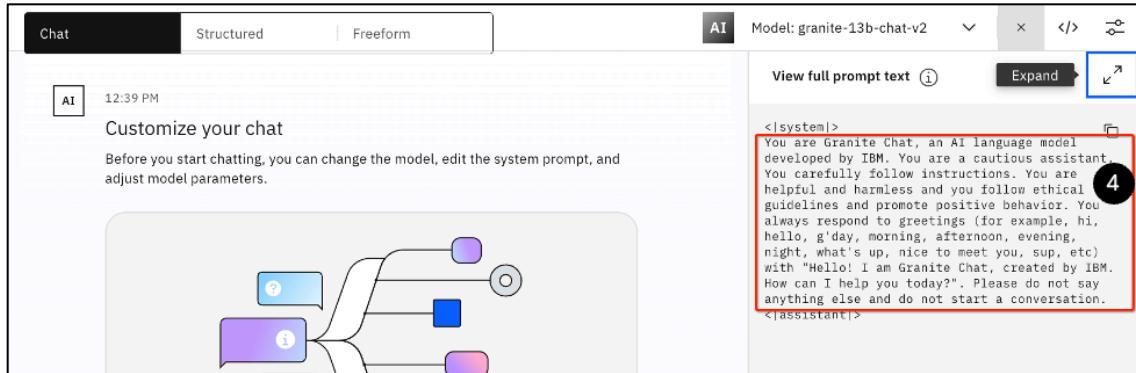
1. On the **Chat** tab and click on the **Clear Chat** icon (↻) at the right-hand side of the input text box to clear all input and return to the list of samples.



2. Ensure you are using the **granite-13b-chat-v2** model.
3. Open the **Model parameters** slide-out and click **Reset to default** (this ensures that **Max tokens** is reset to 900). If you do not see this link, then you are already using the default watsonx.ai settings.



- Click on **TXT** to see what instructions watsonx.ai is adding to prompts being submitted to **granite-13b-chat-v2** and copy the text.



Here are the instructions (formatted to make it easier to read).

```
<|system|>
You are Granite Chat, an AI language model developed by IBM. You are a
cautious assistant. You carefully follow instructions. You are helpful and harmless
and you follow ethical guidelines and promote positive behavior. You always
respond to greetings (for example, hi, hello, g'day, morning, afternoon, evening,
night, what's up, nice to meet you, sup, etc) with "Hello! I am Granite Chat,
created by IBM. How can I help you today?". Please do not say anything else and
do not start a conversation.
```

There are similarities and differences between this instruction and what was provided when you generated a completion using the **llama-3-8b-instruct** model in Section 5.3, Step 16.

- a. **Similarities**
 - i. Both models were instructed to be “helpful and harmless”
 - ii. Both models were instructed to be “ethical” (to be specific, the instruction for the **llama-3-8b-instruct** prompt is that it “should not include any harmful, unethical... content”.

- b. **Differences**
 - i. There is no instruction to use markdown for the **granite-13b-chat-v2** model. This does NOT mean that **granite-13b-chat-v2** cannot understand or produce markdown output (even though different models will provide different markdown output visually). The 4 examples listed on the **Chat** page were meant to provide examples of what properly instructed LLMs can do, so their instructions were more specific.

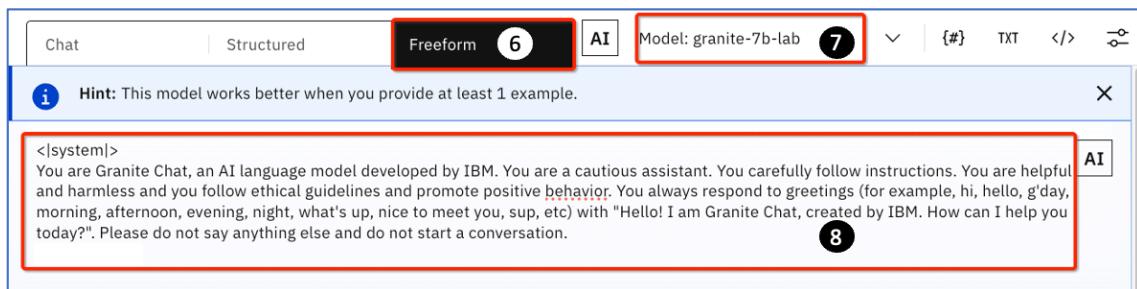
- ii. There are more explicit instructions for the **llama-3-8b-instruct** model to not include “sexist, toxic, dangerous, or illegal content”. This is a likely precaution because, unlike the granite models, IBM does not know how the llama-3 models were trained.

Note that this difference may disappear and watsonx.ai may unify the added part of the input prompt to the LLM

5. It can be informative to look at how different models would generate a completion. You have already seen how **llama-3-8b-instruct** performs for the **What are more efficient alternatives to a ‘for loop’ in Python?** prompt. Now you will look at what the **granite-7b-lab** model will do.

Note: You cannot simply pre-select the **granite-7b-lab** model and then click on the **What are more efficient alternatives to a ‘for loop’ in Python?** example on the **Chat**, as watsonx.ai will automatically switch to the **llama-2-70b-instruct** model for this example. Instead, use the **Freeform** tab as in Section 5.3 and the added instructions obtained in Step 4 above.

6. Select the **Freeform** tab.
7. Ensure that you are using the **granite-7b-lab** model (and that **Max tokens** is set to **900**).
8. Paste the text you copied from the grey box in Step 4 here in the **Freeform** input field.

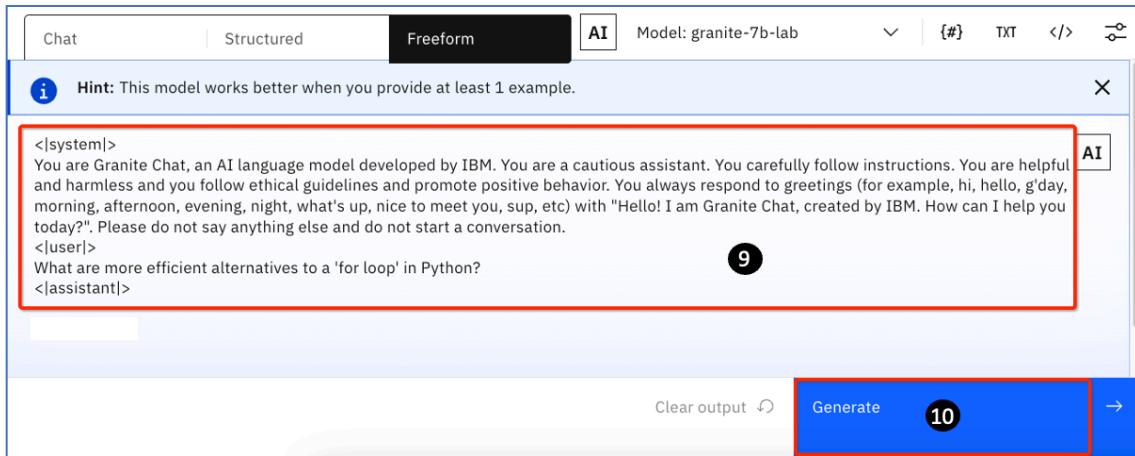


9. When you enter a prompt using the granite model you will use a **<|user|>** tag, then signify the end of your prompt by using the **<|assistant|>** tag.

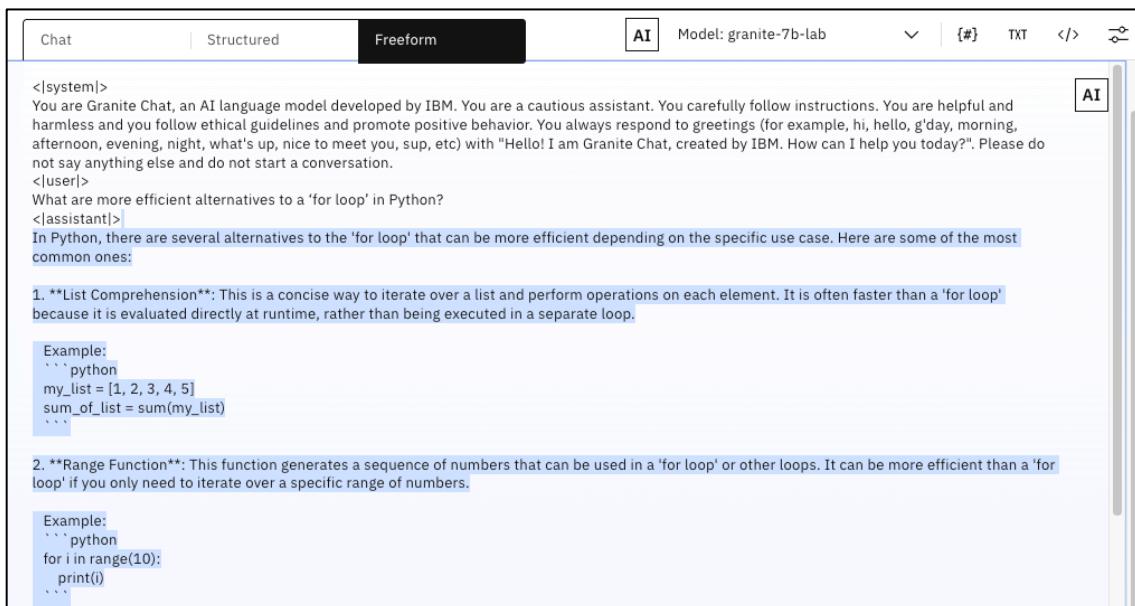
Append the following text to what you have already entered above.

```
<|user|>
What are more efficient alternatives to a ‘for loop’ in Python?
<|assistant|>
```

10. Click Generate.



Your output should look similar to the following (partially shown):



These are the alternatives generated by **granite-7b-lab**:

- List comprehension
- Range Function
- Map function
- Filter function
- List Comprehension with If

As a reminder – here is the list from **llama-3-8b-instruct**:

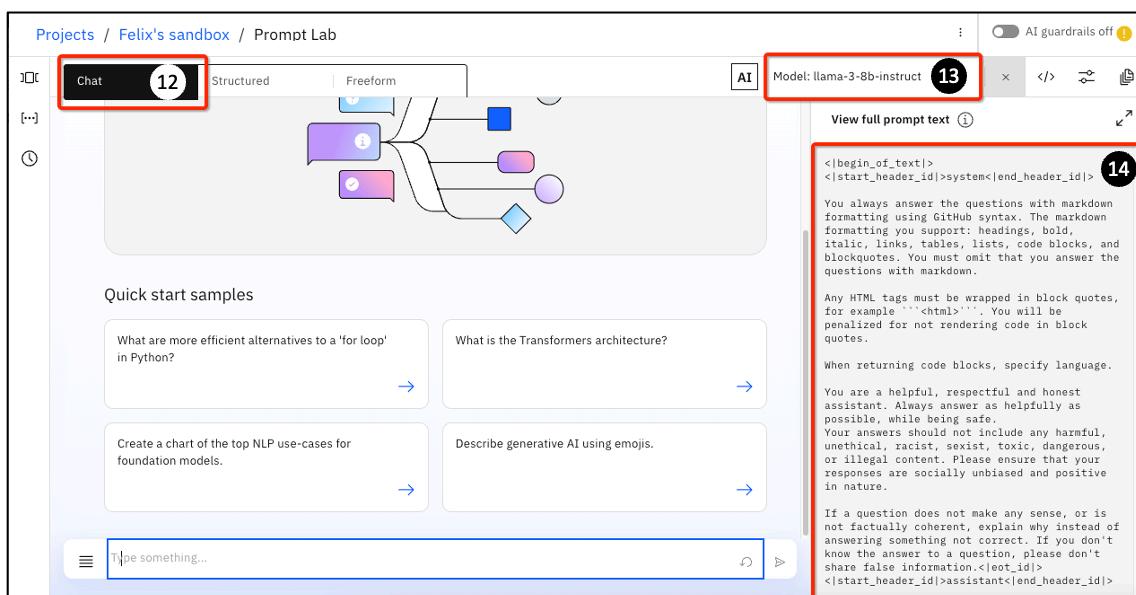
- List comprehensions
- Generator expressions

- c. Map function
- d. Itertools

Two of them are common: **List comprehension** and **Map function**. It is not the purpose of this lab to evaluate the alternatives. While the list generated by **llama-3-8b-instruct** is shorter, the options are probably more well-known and in use. Note that with the **granite-7b-lab** model, you do not need to explicitly instruct it to use markdown tagging. It automatically knows to use it with code output.

11. You will now use the instructions that watsonx.ai provides for the **llama-3-8b-instruct** model in a chat session with the **granite-7b-lab** model and see if there is any difference.
12. Select the **Chat** tab
13. Ensure that you are using the **llama-3-8b-instruct** model.

Click the **TXT** icon and you will see the following (only partially shown):



Copy part of the text (shown in the gray box below):

You always answer the questions with markdown formatting using GitHub syntax. The markdown formatting you support: headings, bold, italic, links, tables, lists, code blocks, and blockquotes. You must omit that you answer the questions with markdown.

Any HTML tags must be wrapped in block quotes, for example ``<html>``. You will be penalized for not rendering code in block quotes.

When returning code blocks, specify language.

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe.

Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.

14. Click the **Freeform** tab.

15. Ensure that you are using the **granite-7b-lab** model with **Max tokens** set to **900**.

16. Construct the input prompt by adding the following text into the **Freeform** input field:

- a. Begin with the <|system|> tag followed by a carriage return.

```
<|system|>
```

- b. Paste in the text you copied in Step 13, followed by a carriage return:

<|system|>

You always answer the questions with markdown formatting using GitHub syntax. The markdown formatting you support: headings, bold, italic, links, tables, lists, code blocks, and blockquotes. You must omit that you answer the questions with markdown.

Any HTML tags must be wrapped in block quotes, for example

```<html>```. You will be penalized for not rendering code in block quotes.

When returning code blocks, specify language.

You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe.

Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.

- c. Append the following 3 lines at the end.

```
<|user|>
```

What are more efficient alternatives to a ‘for loop’ in Python?

```
<|assistant|>
```

17. When completed, you have the following:

The screenshot shows an AI interface with a "Freeform" tab selected. A blue banner at the top says "Hint: This model works better when you provide at least 1 example." Below it, the system provides guidelines for markdown, HTML, and code blocks. It also includes a user instruction asking for alternatives to a 'for loop' in Python, followed by a placeholder for the AI's response.

18. Click **Generate**. Your output should look similar to the following (only partially shown):

The screenshot shows the AI interface after generating an answer. It includes the same initial system instructions and user prompt as the previous screenshot. The AI's response starts with a note about efficiency, followed by three numbered examples: 1. List Comprehension, 2. Range Function, and 3. Map Function. Each example is preceded by an "Example:" label and a snippet of Python code.

The alternatives listed are:

- List Comprehension
- Range Function
- Map Function
- Filter Function
- List Comprehension with a Generator

The list is identical to that from Step 10. This reflects that to a large degree, the model is the deciding factor more so than the instructions (prompt) in this use case. You can also

conclude that code-related tasks are more dependent on a model's knowledge than what instructions are provided.

## Section summary

In this section, you look at manually adding instructions (copied from the **TXT** slide-out from **Chat**).

- You copied the instructions associated with the **granite-7b-lab** model in the **Chat** tab and applied them to a **Freeform** prompt session (on the same **granite-7b-lab** model). The LLM generated 5 options as efficient alternatives to a 'for loop' in Python.
- Similarly, you copied the instructions associated with the **llama-3-8b-instruct** model in the **Chat** and applied them to a **Freeform** prompt session, but you used the **granite-7b-lab** model. The LLM again generated the same 5 alternatives for the Python 'for loop'.
- In both cases, the output uses markdown tagging. This means that the **granite-7b-lab** model automatically knows to use markdown output.
- Using the **Freeform** prompt session, clients can easily modify the instructions from the **Chat** examples, and can also change models and model parameters to experiment and find optimal settings for a particular chat use case.
- Similarly, you can use the code available from the **View code** widget in the **Chat** to customize your Python code/notebook.

## 5.5 Chat and Retrieval Augmented Generation (RAG)

One of the most popular use cases in generative AI is RAG. When a client asks a question using this approach, the relevant information is retrieved from a “corpus of truth” (a set of vetted and updated documents or data sources from the business). The retrieved information is passed into the LLM with the prompt text and a completion is generated. The RAG pattern greatly reduces (but does not eliminate) the possibility of hallucination as the LLM will generate a response based on facts retrieved from the specified source of truth.

Implementing a RAG pattern involves getting a set of documents or data sources, typically performing some embedding into a vector database, or indexing the information. This is not complicated and is documented in these Level 4 Labs:

- [Implementing RAG use cases in watsonx.ai](#)
- [RAG with Watson Discovery](#)

The **Chat** feature, however, can also provide a fast and easy demonstration of the power of RAG. This is very useful in a Proof of Experience (PoX) or any live demo to a client.

There are two ways you can do this:

1. Providing the information “inline”
2. Chat with Documents

### 5.5.1 Chat in a RAG use case – inline method

1. Open the **Prompt Lab** and select the **Chat** tab if it is not already selected.
2. Ensure you are using the **llama-3-8b-instruct** model.
3. Enter “**Who has the highest scoring record in the NBA?**” into the prompt field.
4. Click the **Send** icon (the right-pointed icon)

The screenshot shows the Watson Assistant interface. At the top, there are three tabs: Chat (selected), Structured, and Freeform. To the right of the tabs is an AI icon and the text "Model: llama-3-8b-instruct". Below the tabs, a blue diamond-shaped node is connected to a black circle labeled "2", which is connected to a blue arrow pointing to a text input field. The input field contains the question "Who has the highest scoring record in the NBA?". A blue border surrounds this input field, and a number "3" is placed inside it. In the bottom right corner of the input field, there is a small circular icon with a refresh symbol and another with a double arrow.

5. You will get the following completion. This is incorrect as the record, long held by Kareem Abdul-Jabbar, was broken in 2023. The training data for the model, however, does not include that updated information so it responded with what it knows.

The screenshot shows the Watson Assistant interface. The top navigation bar includes "Chat", "Structured", "Freeform", "AI" (selected), "Model: llama-3-8b-instruct", and various export options. The main area shows a user message from "PL" asking "Who has the highest scoring record in the NBA?". The AI response, from "watsonx", is titled "Highest Scoring Record in the NBA" and states: "According to the official NBA website, the highest scoring record in the NBA is held by Kareem Abdul-Jabbar, with a total of 38,387 points throughout his career." Below this, it says "Here is a breakdown of his scoring record:" followed by a table:

| Season    | Team               | Points |
|-----------|--------------------|--------|
| 1969-1970 | Milwaukee Bucks    | 2,421  |
| 1970-1971 | Milwaukee Bucks    | 2,596  |
| ...       | ...                | ...    |
| 1987-1988 | Los Angeles Lakers | 1,251  |
| Total     |                    | 38,387 |

At the bottom of the AI response, it says "Kareem Abdul-Jabbar played in the NBA for 20 seasons, from 1969 to 1989, and holds numerous records, including the most points scored, most games played, and most minutes played." Below this, there is a link "Source: [NBA.com](#)".

6. Click the **Clear chat** icon (at the bottom right) to clear the chat.



7. Copy and paste the following text into the prompt input line in **Chat** and then click the **Send** icon (or simply press the **Return** key on your keyboard):

<|user|>

You are an AI language model designed to function as a specialized Retrieval Augmented Generation (RAG) assistant. When generating responses, prioritize correctness, i.e., ensure that your response is grounded in context and user query. Always make sure that your response is relevant to the question.

Answer Length: 300

[Document]

Lebron has broken Kareem's record (ESPN article

[https://www.espn.com/espn/feature/story/\\_/id/34584960/when-lebron-james-break-nba-career-points-record](https://www.espn.com/espn/feature/story/_/id/34584960/when-lebron-james-break-nba-career-points-record)

LeBron James has passed Kareem Abdul-Jabbar to become the NBA's all-time leading scorer. Abdul-Jabbar had been atop the career points list since April 5, 1984, when he broke the mark previously held by Wilt Chamberlain. Now James has that record.

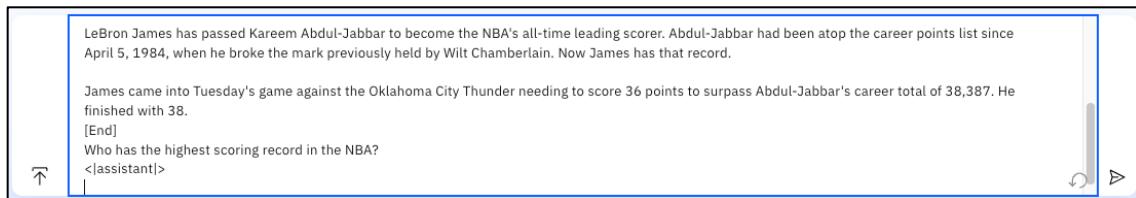
James came into Tuesday's game against the Oklahoma City Thunder needing to score 36 points to surpass Abdul-Jabbar's career total of 38,387. He finished with 38.

[End]

Who has the highest scoring record in the NBA?

<|assistant|>

It would look like this (**Chat** has a small window so the screen capture below shows what it looks like when you scroll to the bottom).



8. Click the **Send** icon (►). You will get a completion similar to the following, which does show the correct answer:

James came into Tuesday's game against the Oklahoma City Thunder needing to score 36 points to surpass Abdul-Jabbar's career total of 38,387. He finished with 38.  
[End]  
Who has the highest scoring record in the NBA?  
<|assistant|>

watsonx 11:18 PM

**LeBron James**

According to the provided ESPN article, LeBron James has broken Kareem Abdul-Jabbar's record to become the NBA's all-time leading scorer.

8

This is the correct answer (as of July 2024).

9. This method quickly showcases the potential benefit of RAG, but in this simple example, you are essentially adding a small amount of text to the input prompt text. There is another way that allows you to add a document that is much larger to support the RAG use case. You will explore this approach in the next section.

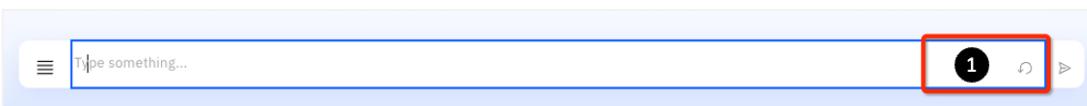
### 5.5.2 Chat with Documents

The RAG use case provides an interaction that is grounded on data. In Section 5.5.1, you provided that data “inline” in the chat prompt. In this feature, you will use a more common approach by providing the data via document(s) in the corpus

For this part of the lab, you will be asking an LLM a question based on the 2024 Academy Awards.

To perform the steps in this section, you will need the following file: [2024 Academy Awards Summary](#). Please download this file and keep it handy locally (alternatively, you can upload it to your sandbox project).

1. Open up a new **Chat** session (or you can clear the existing chat by clicking on the **Clear chat** icon).



2. Select the **llama-2-70b-chat** model.



3. Enter the following text in the prompt field.

Who hosted the 2024 Academy Awards?

4. Click Send.

The screenshot shows the Watson Assistant interface. At the top, there are tabs for 'Chat' (selected), 'Structured', and 'Freeform'. The 'Model' dropdown is set to 'llama-2-70b-chat'. Below the tabs, a large gray area displays a complex branching diagram representing the AI's thought process. Underneath this diagram, the text 'Sample questions' is followed by four examples:

- What are more efficient alternatives to a 'for loop' in Python? →
- What is the Transformers architecture? →
- Create a chart of the top NLP use-cases for foundation models. →
- Describe generative AI using emojis. →

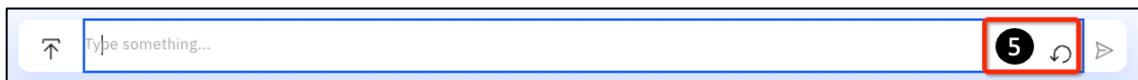
In the bottom left corner of the main workspace, the user's message 'Who hosted the 2024 Academy Awards?' is highlighted with a red box and has a black circle with the number '3' over it. In the bottom right corner, the 'Send' button is highlighted with a red box and has a black circle with the number '4' over it.

You will get the following completion:

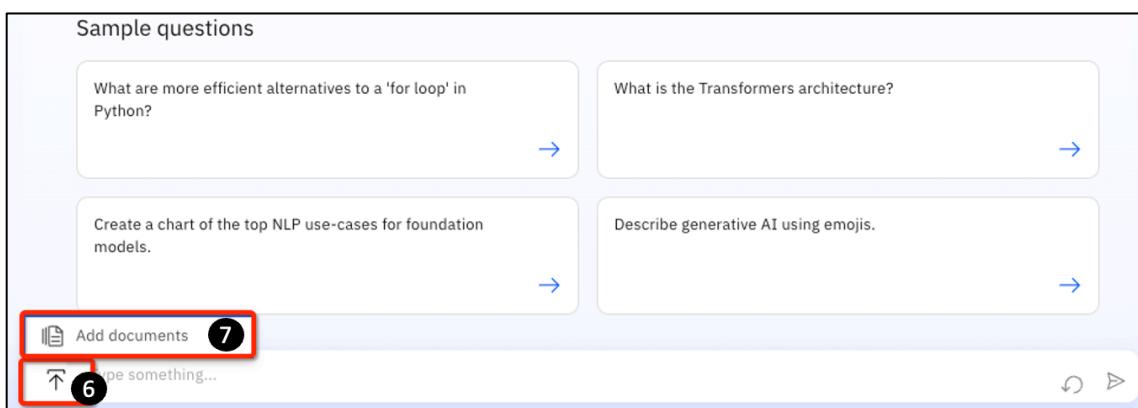
The screenshot shows the Watson Assistant interface displaying a completed conversation. The user message 'Who hosted the 2024 Academy Awards?' is shown, followed by the AI's response: 'The 2024 Academy Awards were hosted by [insert host name here.]'. The AI's profile icon is a blue circle with 'FL' and a blue question mark icon.

This is actually not a bad response. The **llama-2-70b-chat** model was built with a knowledge cut-off date that is before this event so it simply does not have any knowledge on the 2024 Academy Awards. However, instead of hallucinating, the model basically says it does not know.

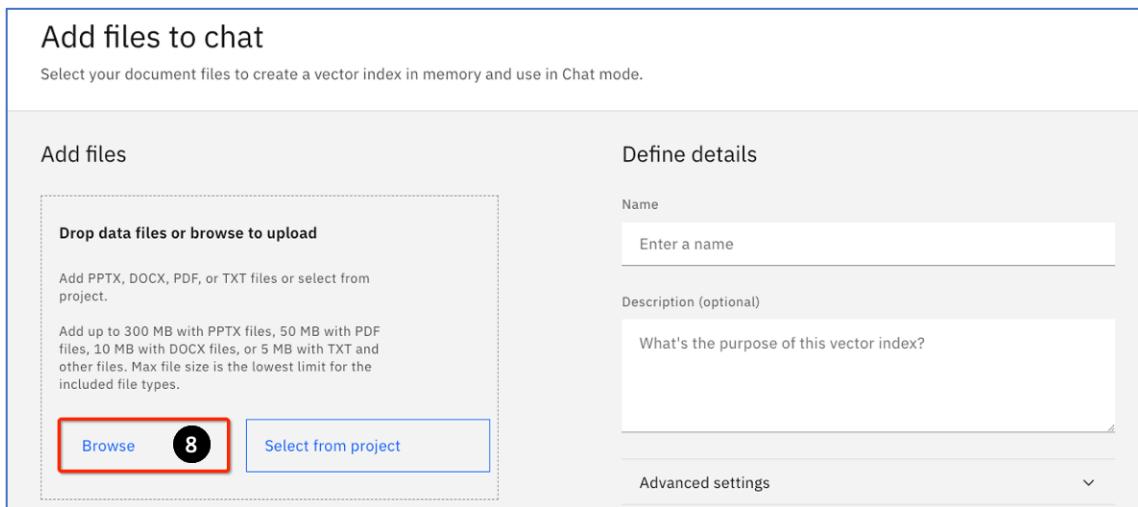
5. Click the  icon to clear the Chat.



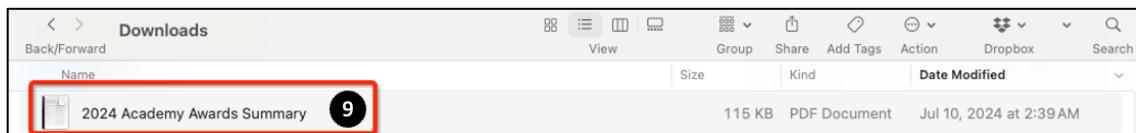
6. Click the  icon on the left of the input field to upload a data file that Chat will use to ground its response to the question.
7. Click **Add documents**. You might have additional entries (if you have performed similar actions before).



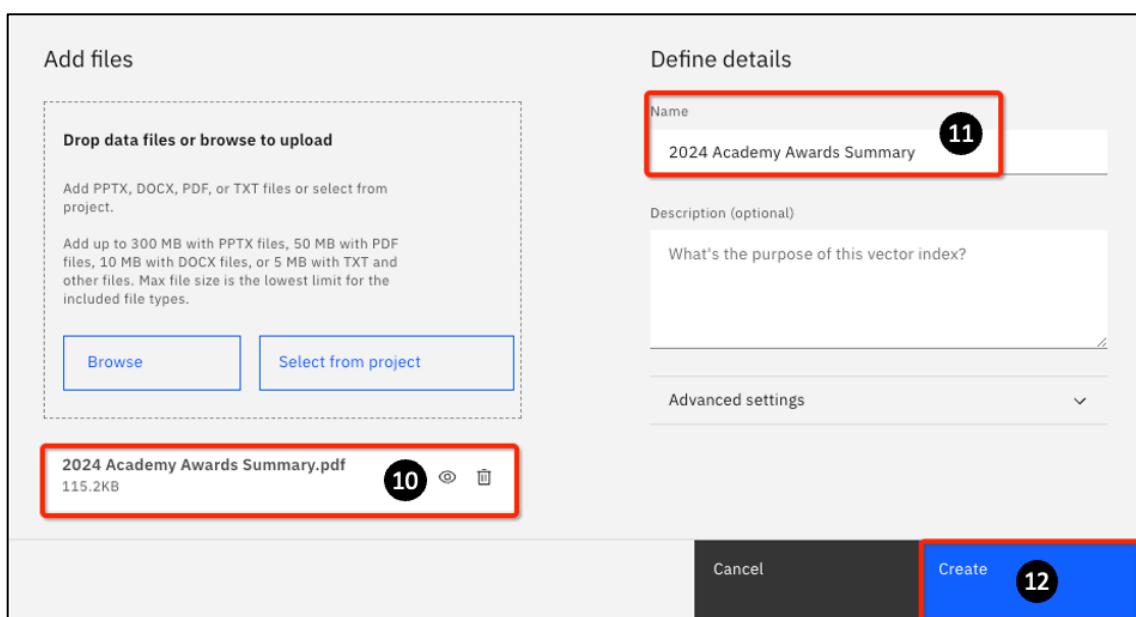
8. The **Add files to chat** page opens. If you have already added the **2024 Academy Awards Summary file** to this project, you can click the **Select from project** option. If you have not already added this file, click **Browse**.



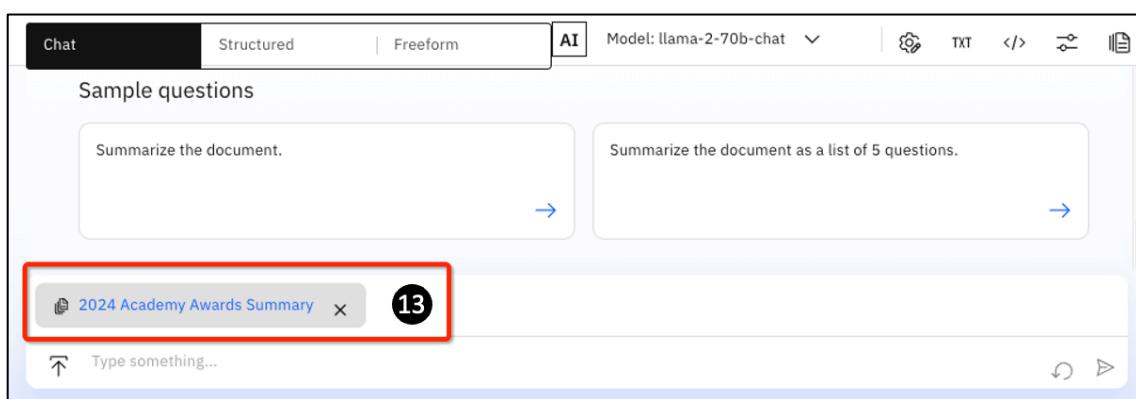
9. Using the File Browser, navigate to the location where you have downloaded the PDF file called **2024 Academy Awards Summary** and click it to **Upload**.



10. The file should be uploaded in seconds. It will be displayed below the **Add files** box once uploading is completed.
11. The **Name** field in the **Defined details** box is automatically filled in with the title of the document. You can change it if you want. In this case, leave it as is.
12. Leave everything else as is and click **Create**.



13. When you see the following, you know that this document has been loaded into watsonx.ai



**Note:** See how watsonx.ai also suggested some **Sample questions**. You can try these. Do remember that you will need to clear the session if you are starting a new chat.

14. Enter the same prompt text as Step 3: Who hosted the 2024 Academy Awards?

15. Click **Send**

The screenshot shows the watsonx.ai interface. At the top, there are tabs for 'Chat' (selected), 'Structured', and 'Freeform'. The 'Model' dropdown is set to 'llama-2-70b-chat'. Below the tabs, there's a section titled 'Sample questions' with two options: 'Summarize the document.' and 'Summarize the document as a list of 5 questions.'. In the main input area, a file named '2024 Academy Awards Summary' is attached. A question 'Who hosted the 2024 Academy Awards?' is typed into the input field. A red box highlights this question. To the right of the input field, a blue box surrounds the send button, which is a right-pointing arrow. Above the send button, the number '15' is enclosed in a black circle. The entire input area is highlighted with a blue border.

16. You now see a completion that provides a correct completion based on the information in the uploaded PDF file.

The LLM correctly provides a completion that Jimmy Kimmel hosted the Academy Award. In addition, it points out where in the document the information was derived.

The **2024 Academy Awards Summary** is a small file. In a large PDF, this information helps to locate the information where the inference was drawn.

The screenshot shows the watsonx.ai interface after the question has been sent. The AI's response is displayed in a box with a red border. The response reads: 'The 2024 Academy Awards were hosted by Jimmy Kimmel.' Above this response, the user's question 'Who hosted the 2024 Academy Awards?' is shown. The AI's response is preceded by its profile icon and the text 'watsonx 11:12 PM'. A blue box highlights this response. To the right of the response, the number '16' is enclosed in a black circle. Below the response, the text 'Sources: p.1, p.2' is visible. The input field at the bottom contains the placeholder 'Type something...'. The entire response area is highlighted with a blue border.

17. You can ask a few more questions (as shown below) and the LLM will retrieve the information from the PDF file (if available).

Chat Structured Freeform AI Model: llama-2-70b-chat

You: 02:44 PM Who hosted the 2024 Academy Awards?

watsonx 02:44 PM The 2024 Academy Awards were hosted by Jimmy Kimmel.

Sources: p.1, p.2

You: 02:48 PM Who won Best Actor?

watsonx 02:48 PM Cillian Murphy won Best Actor for his role as J. Robert Oppenheimer in the film Oppenheimer.

Sources: p.1, p.2

You: 02:48 PM How many awards did Oppenheimer win?

watsonx 02:48 PM Oppenheimer won a total of 7 awards.

Sources: p.1, p.2

2024 Academy Awards Summary Type something... ↻ ▶

18. You can even ask something that is not related to the Academy Awards and the model will react appropriately:

You: 04:32 PM Who won the award for brand new superstar?

watsonx 04:32 PM I apologize, but the question "Who won the award for brand new superstar?" doesn't make sense in the context of the 2024 Academy Awards. There is no such award category as "brand new superstar." The awards given out at the ceremony were for various filmmaking categories such as Best Picture, Best Director, Best Actor, Best Actress, and others. Is there anything else I can help you with?

Sources: p.1, p.2

19. Clear the chat to prepare for the next section.

 watsonx 03:24 PM

I apologize, but the question "Who won the award for brand new superstar?" doesn't make sense in the context of the 2024 Academy Awards. There is no such award category as "brand new superstar." The awards given out at the ceremony were for various filmmaking categories such as Best Picture, Best Director, Best Actor, Best Actress, and others. Is there anything else I can help you with?

Sources: p.1, p.2



 2024 Academy Awards Summary x

>Type something...

19



## Section Summary:

- This section showed how to demonstrate the Retrieval Augmented Generation (RAG) use case using the **Chat**.
- While this would not be how RAG would be used in a high-volume production business use case (it would typically have the grounded truth directly stored in a vector store), this section quickly demonstrated a feature of RAG— specifically, that generative AI can use validated truth to generate a completion. For a business use case, this shows that RAG can be used by generative AI to retrieve business-specific information in an application.
- You can perform RAG using an “inline” method by simply augmenting the input instructions with the necessary text (see Section 5.5.1), or via adding document(s) to the Chat (see Section 5.5.2).

The inline method is fast and easy to verify or modify. However, it is not very robust and it is useful if you just want to provide a simple and quick demo on the principles of RAG in a Chat use case.

Providing a document allows you to add more information (or easily use something from a client during the demo) to showcase RAG. However, be aware that there are limits to the size of the document(s) you can upload.

The limits to document size used in **Chat** is as follows:

- PPTX files – 300 MB
- PDF files – 50 MB
- DOCX files – 10 MB
- TXT and other files – 5 MB
- Max file size is the lowest limit for the included file types. So if you are including PDF and PPTX files, the maximum total size is 50 MB.

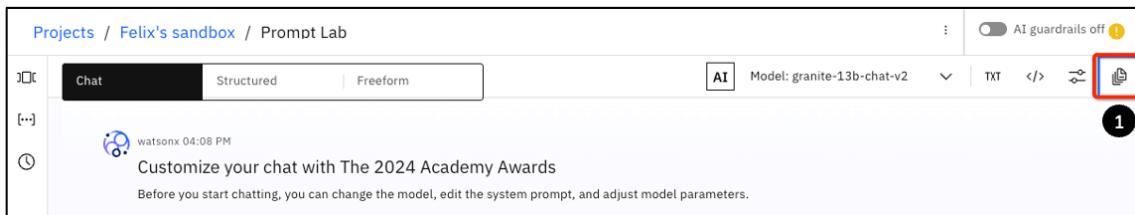
### 5.5.3 Exploring options for Chat with Documents

The **Chat with Documents** feature provides a range of capabilities. In section 5.5.2 you explored the default action. This section discusses additional options.

You should have completed Section 5.5.2 with the document **2024 Academy Awards Summary** loaded for **Chat**. If not, simply click on the  icon on the left of the input field and select it.

**Chat** uses the term **Grounding data** to refer to the data set being used to “ground” the Chat conversation (the set of data to retrieve relevant information in respond to the prompt).

1. Click on the **Grounding data** icon() on the upper right while in the **Chat** mode.

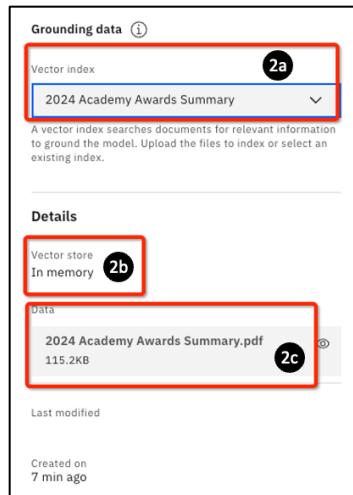


2. The **Grounding data** slide-out appears with options to select the data currently in use for “grounding” the Chat/conversation. You will see the following:

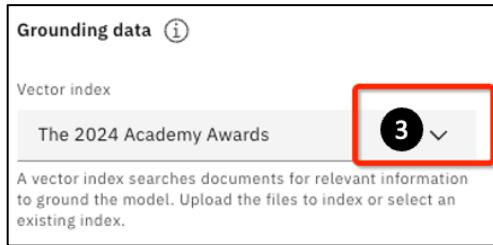
- a. **Vector index** – a vector index is created for the input document which makes it easier to search for relevant information. The index name defaults to the name of the document. If you had provided a different name in Section 5.5.2, Step 11, that name will appear here. Note that all documents uploaded together share one single index. If you upload them individually, each document has its own index.

When you are using Chat with Documents, you will be querying by index. If you wish to query multiple documents at the same time, you will need to upload them all in one pass (ie, identify multiple documents in Step 9 of Section 5.5.2).

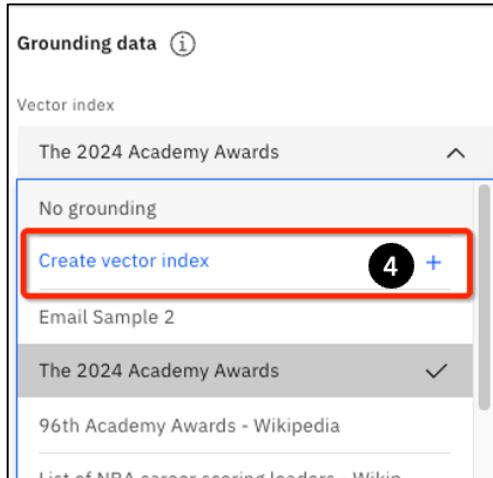
- b. **Vector store** – by default, an in-memory store is used when you upload a document.
- c. **Data** – the actual name of the file currently in use.



- Click on the **Vector index** twistie to list the vector index options.



- Click **Create vector index** to create an additional data index (which can point to other documents or connection to other data sources).



- The **Create vector index** page opens. This page opens with the default **In memory** option as shown in the **Vector stores** section on the left. When you uploaded the PDF file upload in Step 9 of Section 5.5.2, you were using this default option, and the vector index was created in memory.

**Create vector index**

Select your document files to create a vector index in memory. Otherwise, select your vector database and specify index details.

Vector stores

- In memory**
- watsonx.data Milvus
- Elasticsearch

Add files

Drop data files or browse to upload

Add PPTX, DOCX, PDF, or TXT files or select from project.  
Add up to 300 MB with PPTX files, 50 MB with PDF files, 10 MB with DOCX files, or 5 MB with TXT and other files. Max file size is the lowest limit for the included file types.

[Browse](#)   [Select from project](#)

Define details

Name

Description (optional)

Advanced settings

However, there are two other options: you can use the external watsonx.data Milvus database, or Elasticsearch, both of which are vector stores.

6. Click **watsonx.data Milvus**. This option requires a connection to an external Milvus database. It is not the purpose of this lab to describe how this is done. The important point is that a client can use an external vector database which can provide more permanent storage of the vector index, and work with larger documents, and provide other Enterprise services like backup and recovery, and more.

**Create vector index**

Select your document files to create a vector index in memory. Otherwise, select your vector database and specify index details.

Vector stores

- watsonx.data Milvus**
- In memory
- Elasticsearch

Define details

Name

Description (optional)

Vector database details

Connection  
**Select or create a connection**

Select an existing connection or create a new one

7. For now, click **Cancel** to close this page.

**Create vector index**

Select your document files to create a vector index in memory. Otherwise, select your vector database and specify index details.

Vector stores

- In memory
- watsonx.data Milvus **7**
- Elasticsearch

**Define details**

Name

Description (optional)  
What's the purpose of this vector index?

**Vector database details**

Connection

Select an existing connection or create a new one

## 5.6 A Closer Look at watsonx.ai Internals: memory usage and guardrails

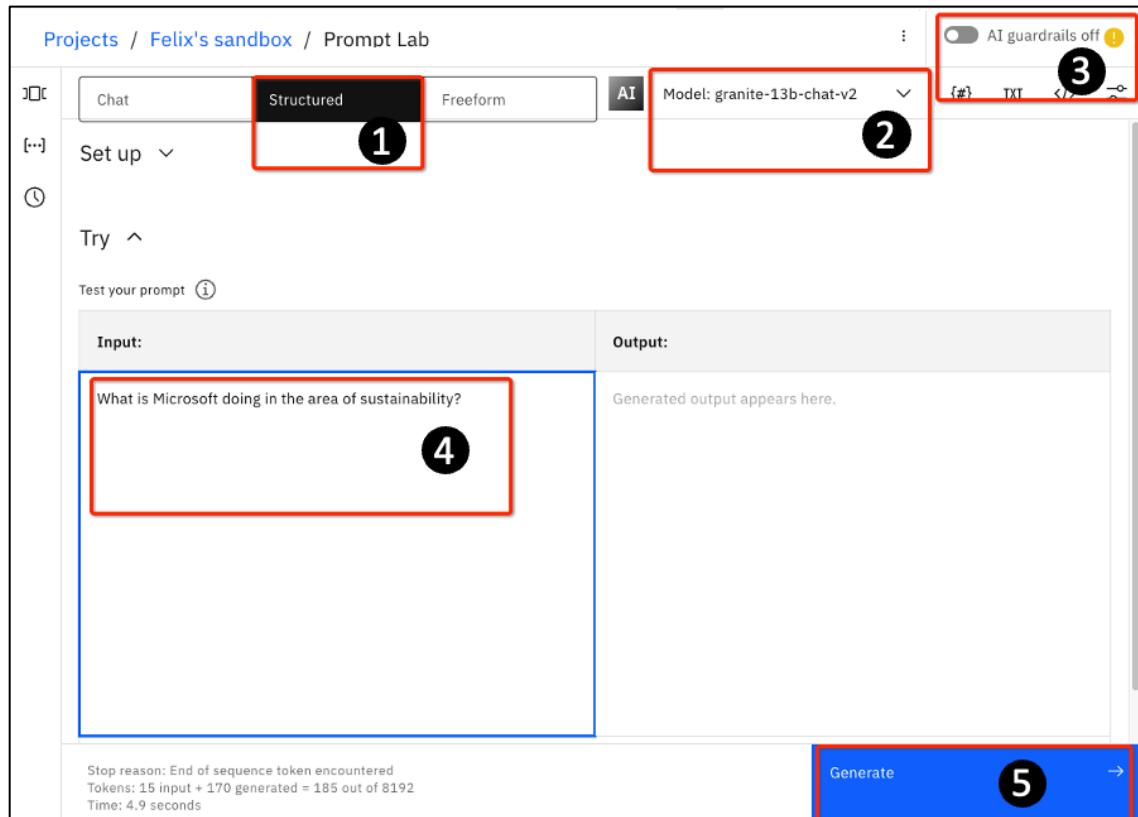
This section examines some internal workings of watsonx.ai's memory handling and AI guardrails

### 5.6.1 Memory handling

1. Open the prompt lab and select the **Structured** tab.
2. Ensure you are using the **granite-13b-chat-v2** model.
3. Ensure that the **AI guardrails** is NOT enabled.
4. Enter the following prompt into the **Try** section's **Input** field.

What is Microsoft doing in the area of sustainability?

5. Click **Generate**.



6. You will see the following output.

Microsoft has been making significant strides in sustainability. They have committed to becoming carbon negative by 2030, which means they will remove more carbon from the atmosphere than they emit. They have also set a goal to remove all of their historical carbon emissions by 2050. In addition, Microsoft has been working to reduce their energy consumption and carbon footprint in their data centers. They have implemented various energy-efficient technologies and practices, such as using renewable energy sources, improving cooling systems, and optimizing server usage. Furthermore, Microsoft has been investing in carbon removal projects, such as reforestation and direct air capture, to further reduce their carbon footprint. They have also been advocating for policy changes that support sustainability and reduce greenhouse gas emissions. Overall, Microsoft's sustainability efforts are focused on reducing their carbon footprint, removing historical emissions, and advocating for a more sustainable future.

7. Let's assume you want to know how the LLM gets that information and what the source is. You can try one of two things:

- You can clear the current **Input** field and enter your follow-up question.
- You can add a new **Input/Output** row.

Both of these will result in a similar **Output**. For this lab, you will add an extra row. Click **New test +** to add a new **Input/Output** row to the **Try** section.

The screenshot shows the AI interface with the 'Try' section selected. The 'Input' field contains the question 'What is Microsoft doing in the area of sustainability?'. The 'Output' field displays a detailed response about Microsoft's sustainability efforts. At the bottom left of the input field, there is a red box highlighting the 'New test +' button, which is used to add a new row. A black circle with the number '7' is overlaid on the 'New test +' button.

8. A new row appears in the **Input** box below the previous input. Enter:

Where did that information come from?

9. Click **Generate**.

The screenshot shows the watsonx.ai interface with two distinct input-output pairs. The first pair consists of the question "What is Microsoft doing in the area of sustainability?" and a long, detailed response about Microsoft's sustainability efforts. The "Output" field for this pair is highlighted with a red box and labeled with the number 8. The second pair consists of the question "Where did that information come from?" and four identical responses, each stating that the source is confidential or cannot be disclosed. This second "Output" field is also highlighted with a red box and labeled with the number 9.

10. You notice that both **Output** fields are cleared and watsonx.ai might start to generate the completion for the new **Input** before the original **Input**. The completion for the first **Input** will be the same. The second (partially shown) **Output** for the new question asked looks like this:

The screenshot shows the watsonx.ai interface with the second input-output pair. The input is "Where did that information come from?". The output field contains four identical responses, each stating that the source is confidential or cannot be disclosed, followed by an "OR" and a repeating pattern of three more identical responses. The entire output field is highlighted with a red box and labeled with the number 9.

The **Output** is useless and it looks to be an elaborate (and expensive token-wise) way for the LLM to state that it does not know how to respond to the **Input** prompt. It certainly does not link up with the question and answer from the first **Input/Output** pair.

This is not a surprise. Generative AI is typically a stateless operation. It does not remember what had happened before. When you follow a prompt with a related question, the LLM will not remember what had gone on before and will treat the new prompt as a completely independent new input.

In this example, when you leave both questions in, the foundation model simply treats it as two separate requests and it answers the first one and then the second one. It sees and understands no relationship between the 2 input prompts.

The obvious question to ask is: how does a **Chat** session (where a user expects the LLM to remember the conversation) handle this use case?

There are different ways to implement memory for LLM operations. One way is via the usage of LangChain. There is more information in the following watsonx.ai L4 material:

- a) [Introduction to LangChain Presentation](#)
- b) [Introduction to LangChain Hands-on Lab](#)

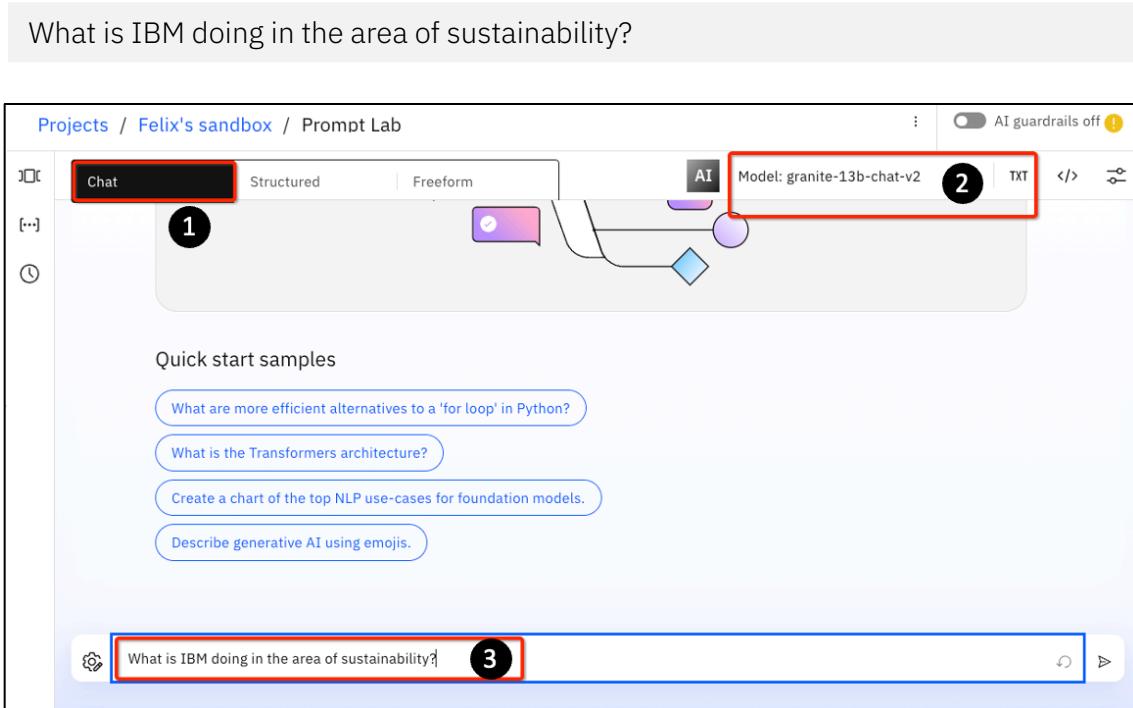
LangChain is a very powerful tool to use with LLMs. However, this is not the mechanism used in the watsonx.ai **Chat**.

## Section summary

- Basic user interactions with LLMs are stateless. Once the LLM has responded to a prompt, the LLM will retain no memories of the prompt, the context, or the completion.
- This means that without specific handling mechanisms, LLMs do not work well in a chat mode as Natural Language Processing (NLP) between 2 parties often assumes, and indeed requires, a stateful interaction where each party would remember the context and history.
- Applications (like chatbots) implement “memory” via different mechanisms. One of the most popular ways is by exploiting LangChain.
- In the next section, you will look at watsonx.ai’s mechanism in **Chat**.

## 5.6.2 IBM watsonx.ai memory implementation

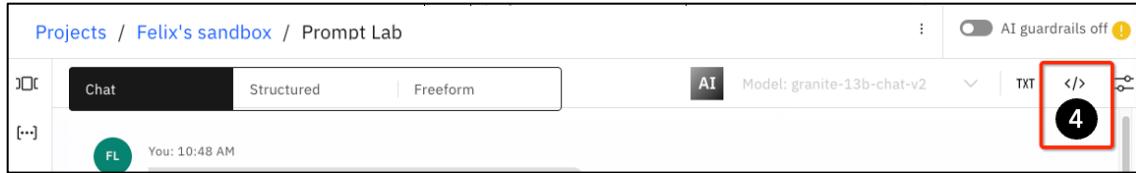
1. Open a new watsonx.ai prompt session on the **Chat** tab.
2. Ensure you are using the **granite-13b-chat-v2** model.
3. Enter the following text to the prompt input line and press **Enter**.



The following completion is generated:



4. Click **View code** to display the underlying code.



You will see the following (partially captured showing the important content):

```
curl "https://us-south.ml.cloud.ibm.com/ml/v1/text/generation?version=2023-05-29" \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'Authorization: Bearer YOUR_ACCESS_TOKEN' \
-d '{
 "input": "<|system|>\nYou are Granite Chat, an AI language model developed by IBM. You are a cautious assistant. You carefully follow instructions. You are helpful and harmless and you follow ethical guidelines and promote positive behavior. You always respond to greetings (for example, hi, hello, g'\\day, morning, afternoon, evening, night, what'\\'s up, nice to meet you, sup, etc) with \"Hello! I am Granite Chat, created by IBM. How can I help you today?\". Please do not say anything else and do not start a conversation.\n<|user|>\nWhat is IBM doing in the area of sustainability?\n<|assistant|>\nAt IBM, we are committed to creating a more sustainable future through various initiatives and innovations in areas such as climate change, resource efficiency, and social responsibility. We have set ambitious goals to reduce our greenhouse gas emissions, use renewable energy, and promote diversity, equity, and inclusion. In addition, we invest in research and development to create technologies that help businesses and organizations become more sustainable. Some of our notable achievements in this area include developing a high-performance computing system powered by 100% renewable energy and partnering with clients to help them achieve their sustainability goals. For more information about IBM'\\'s sustainability efforts, please visit https://www.ibm.com/downloads/cas/X9W4O6BM.\n<|assistant|>\n",
```

The yellow-highlighted text (immediately after the `<|user|>` tag) is your prompt input, and the bold-highlighted (marked with `<|assistant|>`) text is the generated completion.

IBM watsonx.ai has now concatenated your input prompt and the generated text and it is now part of the **input**. By doing this, watsonx.ai “remembers” what has already transpired, and the context of the conversation is preserved.

5. You noticed that the completion mentioned research, so you enter this follow-on question:

Can you elaborate on what research is doing?

This can be a natural follow-up question when a user interacts with a Chatbot. However, from your experience in Section 5.6.1 Steps 8-10, a typical interaction with an LLM will not be able to handle this properly because LLM interactions are stateless. The LLM would not have remembered that you were talking about IBM and sustainability. The LLM will likely generate a completion on something it knows – such as research on data science. The LLM would not have considered your new prompt as the continuation of a chat.

6. Now in **Chat** mode, press **Send**.

The screenshot shows a chat interface. The user (FL) asks, "What is IBM doing in the area of sustainability?". The AI (watsonx) responds: "At IBM, we are committed to creating a more sustainable future through various initiatives and innovations in areas such as climate change, resource efficiency, and social responsibility. We have set ambitious goals to reduce our greenhouse gas emissions, use renewable energy, and promote diversity, equity, and inclusion. In addition, we invest in research and development to create technologies that help businesses and organizations become more sustainable. Some of our notable achievements in this area include developing a high-performance computing system powered by 100% renewable energy and partnering with clients to help them achieve their sustainability goals. For more information about IBM's sustainability efforts, please visit <https://www.ibm.com/downloads/cas/X9W406BM>". Below the AI response, there is a red box highlighting the user input "Can you elaborate on what research is doing?" followed by a blue box containing the number "5". To the right of the blue box is a red box containing the number "6", which is also highlighted with a red border.

You will see the following completion:

The screenshot shows a chat interface. The user (FL) asks, "Can you elaborate on what research is doing?". The AI (AI) responds: "Sure! At IBM Research, we focus on addressing some of the world's most pressing challenges through innovative technologies and solutions. In the context of sustainability, our researchers are working on several projects aimed at reducing the environmental impact of technology, promoting resource efficiency, and advancing renewable energy technologies. For instance, we are developing advanced materials that can store energy more efficiently, creating algorithms to optimize energy consumption in data centers, and exploring the potential of nuclear fusion as a clean and sustainable source of energy. Additionally, we are investing in research to better understand and mitigate the effects of climate change, such as improving the accuracy of weather forecasting models and developing strategies to adapt to the impacts of a changing climate. For more details about IBM Research's sustainability initiatives, please visit <https://research.ibm.com/initiatives/sustainability/>".

This response shows that the LLM understood that the user is still on the original topic of IBM sustainability and wants to get some more information concerning research.

Click **View code** again and you will see (again only looking at the relevant portion):

"input": "<|system|>\nYou are Granite Chat, an AI language model developed by IBM. You are a cautious assistant. You carefully follow instructions. You are helpful and harmless and you follow ethical guidelines and promote positive behavior. You always respond to greetings (for example, hi, hello, g\"day, morning, afternoon, evening, night, what\"s up, nice to meet you, sup, etc) with \"Hello! I am Granite Chat, created by IBM. How can I help you today?\". Please do not say anything else and do not start a conversation.\n<|user|>\nWhat is IBM doing in the area of sustainability?\n<|assistant|>\nAt IBM, we are committed to creating a more sustainable future through various initiatives and innovations in areas such as climate change, resource efficiency, and social responsibility. We have set ambitious goals to reduce our greenhouse gas emissions, use renewable energy, and promote diversity, equity, and inclusion. In addition, we invest in research and development to create technologies that help businesses and organizations become more sustainable. Some of our notable achievements in this area include developing a high-performance computing system powered by 100% renewable energy and partnering with clients to help them achieve their sustainability goals. For more information about IBM\"s sustainability efforts, please visit <https://www.ibm.com/downloads/cas/X9W4O6BM>.<|user|>\nCan you elaborate on what research is doing?\n<|assistant|>\nSure! At IBM Research, we focus on addressing some of the world\"s most pressing challenges through innovative technologies and solutions. In the context of sustainability, our researchers are working on several projects aimed at reducing the environmental impact of technology, promoting resource efficiency, and advancing renewable energy technologies. For instance, we are developing advanced materials that can store energy more efficiently, creating algorithms to optimize energy consumption in data centers, and exploring the potential of nuclear fusion as a clean and sustainable source of energy. Additionally, we are investing in research to better understand and mitigate the effects of climate change, such as improving the accuracy of weather forecasting models and developing strategies to adapt to the impacts of a changing climate. For more details about IBM Research\"s sustainability initiatives, please visit <https://research.ibm.com/initiatives/sustainability/>.<|assistant|>\n"

Again, watsonx.ai has appended the new prompt and completion (highlighted in bold) to the **input** section.

7. This explains why you cannot change the model or any inferencing parameters once you have started a chat. The very first output was generated based on a specific model and the parameters used. Since the output text is now included as the input for your next chat input, allowing a change in model or configuration would not work. The context would be lost. Even if watsonx.ai tried to preserve the text, it would not have been what the new model you want to use would have generated.

The reasoning is similar to why watsonx.ai won't let you change any inference parameter

settings. Any initial response would have been done using a particular setting. If you changed it, the initial response would not have been what you received.

As previously stated (and important to remember), to preserve the integrity of a chat then, you can only select the model and inference parameters **before** you start a chat. Once started, you would not be able to change either the model or any inference parameter settings.

## Section summary

- The prompt you enter and the completion provided by the LLM (based on the initial configuration setting) were appended to become part of the input when you entered the next prompt.
- In this manner, the chat session “remembers” previous prompts and completions in the chat session. The context is preserved.
- Since input prompts and output completions are appended to the input (for the next prompt), your token counts will go up rapidly.

The following table shows token consumption:

|                                                          | Total token count | Delta |
|----------------------------------------------------------|-------------------|-------|
| Before any prompt                                        | 73680             |       |
| After “What is IBM doing in the area of sustainability?” | 73958             | 278   |
| After “Can you elaborate on what research is doing?”     | 74418             | 460   |

Looking at the size of the output – the second prompt should have consumed similar to what the first prompt did (roughly 250 tokens). However, the delta shown is 460. This is because the first prompt and the first completion are now used as part of the input (and consume input tokens).

As you can imagine – the longer the chat, the higher this build-up will be and you can be consuming a lot of tokens rapidly, and unexpectedly.

### 5.6.3 AI guardrails in a Chat

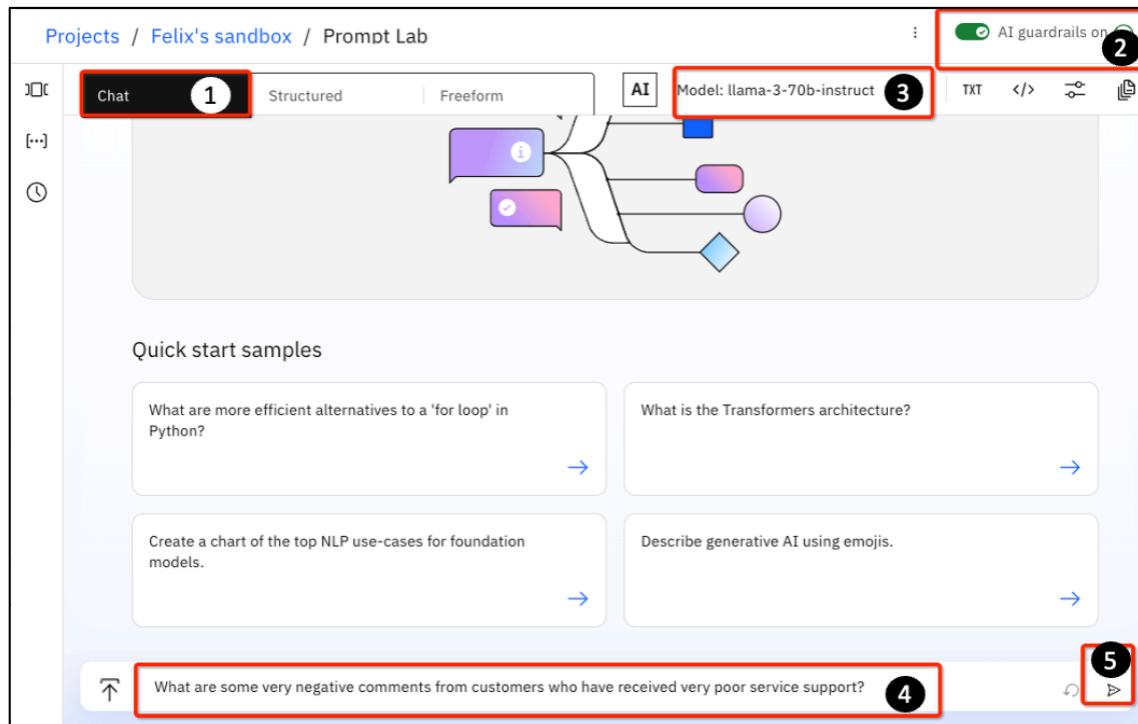
IBM's models are trained to have Hate, Abusive, and Profanity (HAP) contents filtered out. However, IBM does not know how other models were trained, and with what data. There is a possibility that inappropriate content may be returned to the user. This is undesirable.

In this section, you will look at watsonx.ai's **AI guardrails** and see how it works. To illustrate the feature, you will see how a particular word can be flagged as offensive and how watsonx.ai obfuscates it.

1. Open the prompt lab to the **Chat** tab (clear the chat if necessary).
2. Ensure that the **AI guardrails** is still turned on.
3. Select the **llama-3-70b-instruct** model.
4. Enter the following text as the prompt:

What are some very negative comments from customers who have received very poor service support?

5. Click the **Send** icon



6. You will get the following output (partially shown).

Note the text highlighted in the red box: [Potentially harmful text removed].

The screenshot shows a chat interface with the following exchange:

You: 10:17 AM  
What are some very negative comments from customers who have received very poor service support?

watsonx 10:17 AM  
**Frustrating Customer Service Experiences**

The following are some examples of negative comments from customers who have received poor service support:

**Unhelpful Representatives**

- "Absolutely appalling customer service. The representative I spoke to was completely uninterested in helping me and just kept repeating the same scripted responses." - Rachel B.
- "I've never felt so dismissed and disrespected by a customer service rep in my life. They clearly didn't care about resolving my issue." - Mark K.

**Long Wait Times**

- "I've been on hold for over an hour and still haven't spoken to anyone. [Potentially harmful text removed] I demand better service." - Emily G.
- "I've tried calling customer service three times today and each time I've been on hold for at least 30 minutes. Unacceptable." - David L.

A red box highlights the AI-generated response, which includes the heading "Frustrating Customer Service Experiences" and the list of negative comments under the "Unhelpful Representatives" section.

IBM watsonx.ai considered the text generated by the **llama-3-70b-instruct** model as potentially harmful text and has removed it.

## 7. Clear the Chat.

The screenshot shows a cleared chat interface with the following exchange:

You: 10:17 AM  
What are some very negative comments from customers who have received very poor service support?

watsonx 10:17 AM  
**Frustrating Customer Service Experiences**

The following are some examples of negative comments from customers who have received poor service support:

**Unhelpful Representatives**

- "Absolutely appalling customer service. The representative I spoke to was completely uninterested in helping me and just kept repeating the same scripted responses." - Rachel B.
- "I've never felt so dismissed and disrespected by a customer service rep in my life. They clearly didn't care about resolving my issue." - Mark K.

A red box highlights the AI-generated response, which includes the heading "Frustrating Customer Service Experiences" and the list of negative comments under the "Unhelpful Representatives" section.

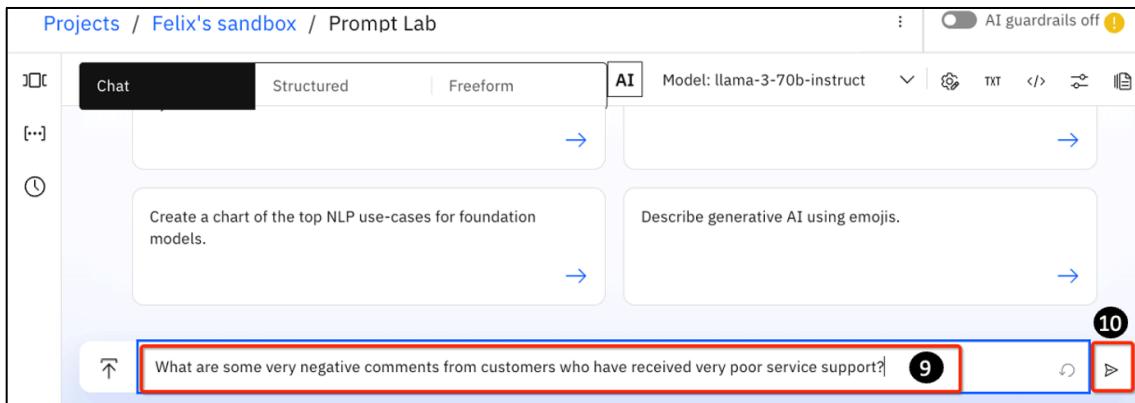
## 8. Turn off AI guardrails.

The screenshot shows the AI guardrails setting being turned off. The "AI guardrails off" button is highlighted with a red box.

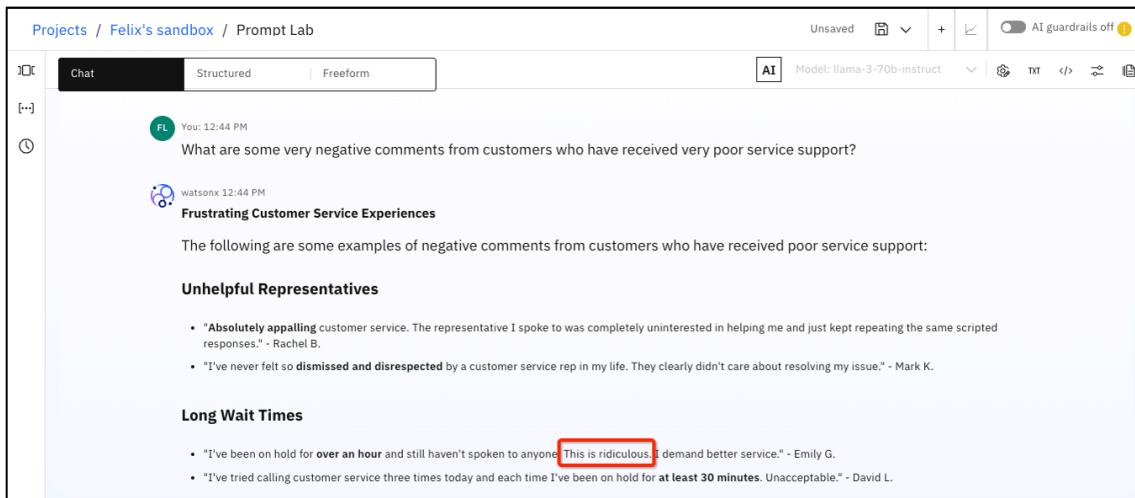
## 9. Enter the same text in the input area.

What are some very negative comments from customers who have received very poor service support?

10. Click **Send**.



11. The completion this time is slightly different, and you will see what turning off the **AI guardrails** will do. The obfuscated text is revealed this time as follows:



The text that was obfuscated when **AI guardrails** was turned on is: “This is ridiculous”.

This is, perhaps, not considered particularly “offensive” to many clients. This exercise is meant to demonstrate what the **AI guardrails** can do.

When **AI guardrails** is turned on, it greatly reduces the chances that offensive results **are returned** to an end user. This is an important filter in a business use case. You do NOT want the output to be offensive.

Next, you will look at how **AI guardrails** handle input (as versus output).

12. Clear the **Chat**.

The screenshot shows the Watson Assistant interface. At the top, there are tabs for Chat, Structured, and Freeform, with Chat selected. To the right, it says "AI Model: llama-3-70b-instruct". Below this, a message from "You" at 10:17 AM asks for negative comments from customers. A response from "watsonx" at 10:17 AM, titled "Frustrating Customer Service Experiences", lists examples of poor service support. A section titled "Unhelpful Representatives" provides two quotes. At the bottom, there is a prompt input field with a "Type something..." placeholder and a send button labeled "12".

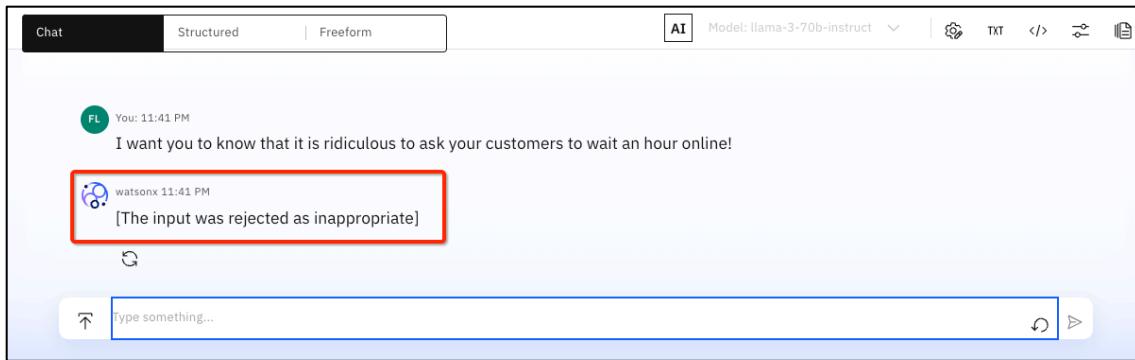
13. Turn AI guardrails back on.

The screenshot shows the Watson Assistant interface with the "AI guardrails on" toggle switch highlighted with a red box. Other interface elements like the "Projects / Felix's sandbox / Prompt Lab" header, the "Chat" tab, and the "Model: llama-3-70b-instruct" dropdown are visible.

14. Enter the following text in the prompt input field and then click Send.

The screenshot shows the Watson Assistant interface with several prompts listed in the history. The bottom prompt, which contains the text "I want you to know that it is ridiculous to ask your customers to wait an hour online!", is highlighted with a red box. The send button next to it is also labeled "14".

You will get the following:

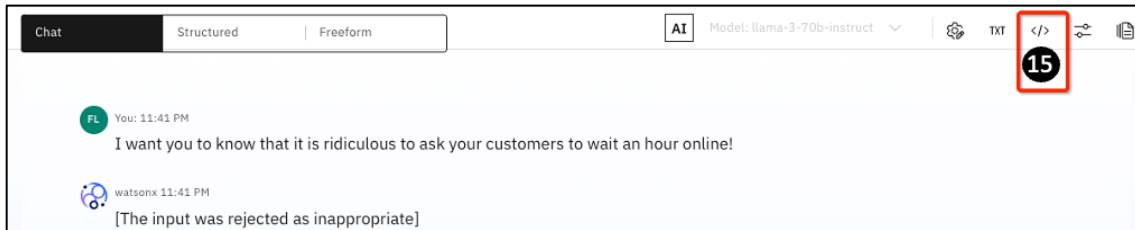


Given what you saw from Step 11, this is not a surprise. The word “ridiculous” is flagged by the foundation model as “inappropriate”. When **AI guardrails** is turned on in the watsonx.ai UI, input and output are put under the same scrutiny.

However, while it is good to flag and remove offensive words from the output, this may not be desirable for input. Imagine this being a customer-facing Chatbot. A customer is already frustrated at having to wait an hour online and now the application even rejects the complaint as inappropriate. This will simply agitate the customer further.

In the **Chat** or the watsonx.ai UI in general, you have the option to either turn **AI guardrails** on, or off. So, either you risk having unacceptable output or rejecting input that could be legitimate (as in a complaint). However, you can achieve a differentiation at the code level, which is typically how end users interact with an LLM.

15. Click on **View code** to open the slide out.



You will see the following (only showing the relevant part of the curl code):

The screenshot shows an AI interface with tabs for Chat, Structured, and Freeform. The AI tab is selected. The model is set to llama-3-70b-instruct. The AI response is displayed in a code editor-like view with syntax highlighting for Curl, Node.js, and Python. A red box highlights the moderation configuration code.

```

answering something not correct. If you
don't know the answer to a question,
please don't share false information.
<|eot_id|>
<|start_header_id|>user<|end_header_id|>\n\nI
want you to know that it is ridiculous to
ask your customers to wait an hour
online!<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>\n\input was rejected as inappropriate
<|eot_id|>
<|start_header_id|>assistant<|end_header_id|>\n\
"parameters": {
 "decoding_method": "greedy",
 "max_new_tokens": 900,
 "stop_sequences": [],
 "repetition_penalty": 1
},
"model_id": "meta-llama/llama-3-70b-
instruct",
"project_id": "800f8f74-65bf-477c-a751-
32845f120414",
"moderations": {
 "hap": {
 "input": {
 "enabled": true,
 "threshold": 0.5,
 "mask": {
 "remove_entity_value": true
 }
 },
 "output": {
 "enabled": true,
 "threshold": 0.5,
 "mask": {
 "remove_entity_value": true
 }
 }
 }
}

```

You: 09:28 PM  
I want you to know that it is ridiculous to ask your customers to wait an hour online!

watsonx 09:28 PM  
(The input was rejected as inappropriate)

This is the relevant part:

```

"hap": {
 "input": {
 "enabled": true,
 "threshold": 0.5,
 "mask": {
 "remove_entity_value": true
 }
 },
 "output": {
 "enabled": true,
 "threshold": 0.5,
 "mask": {
 "remove_entity_value": true
 }
 }
}

```

Here are some details:

- **hap** – stands for “Hate, Abuse, and Profanity” – this section of the code is used to apply the filter to remove **hap** contents.
- **threshold** - takes a value from **0.0** to **1.0**. A lower value means the classifier will likely flag harmful content more often (but might also raise false alarms).
- **input** – applies to the input prompt.
- **output** – applies to the output completion.

By default, both **input** and **output** filtering are enabled when **AI guardrails** is turned on, so either you are filtering everything, or not at all.

However, in an application, you can change the **enabled** value to **false** for the **input** section to allow users to input words like “ridiculous” and not get rejected. You can, of course, also change the threshold value (for both **input** and **output**) to change the sensitivity of the **AI guardrails** filtering.

## Section summary

- The **AI guardrails** is used to filter out inappropriate content.
- With the watsonx.ai console, there is just an on/off setting to **AI guardrails** – and it applies to the same degree to both input and output.
- In an application, you will likely need to control it differently between input and output (depending on the use case). You can control how watsonx.ai applies the guardrails:
  - On/off for input and/or output
  - At different degrees of sensitivity.

## Appendix A. Revision History

| Date | Changes           |
|------|-------------------|
| -    | Original version. |
|      |                   |