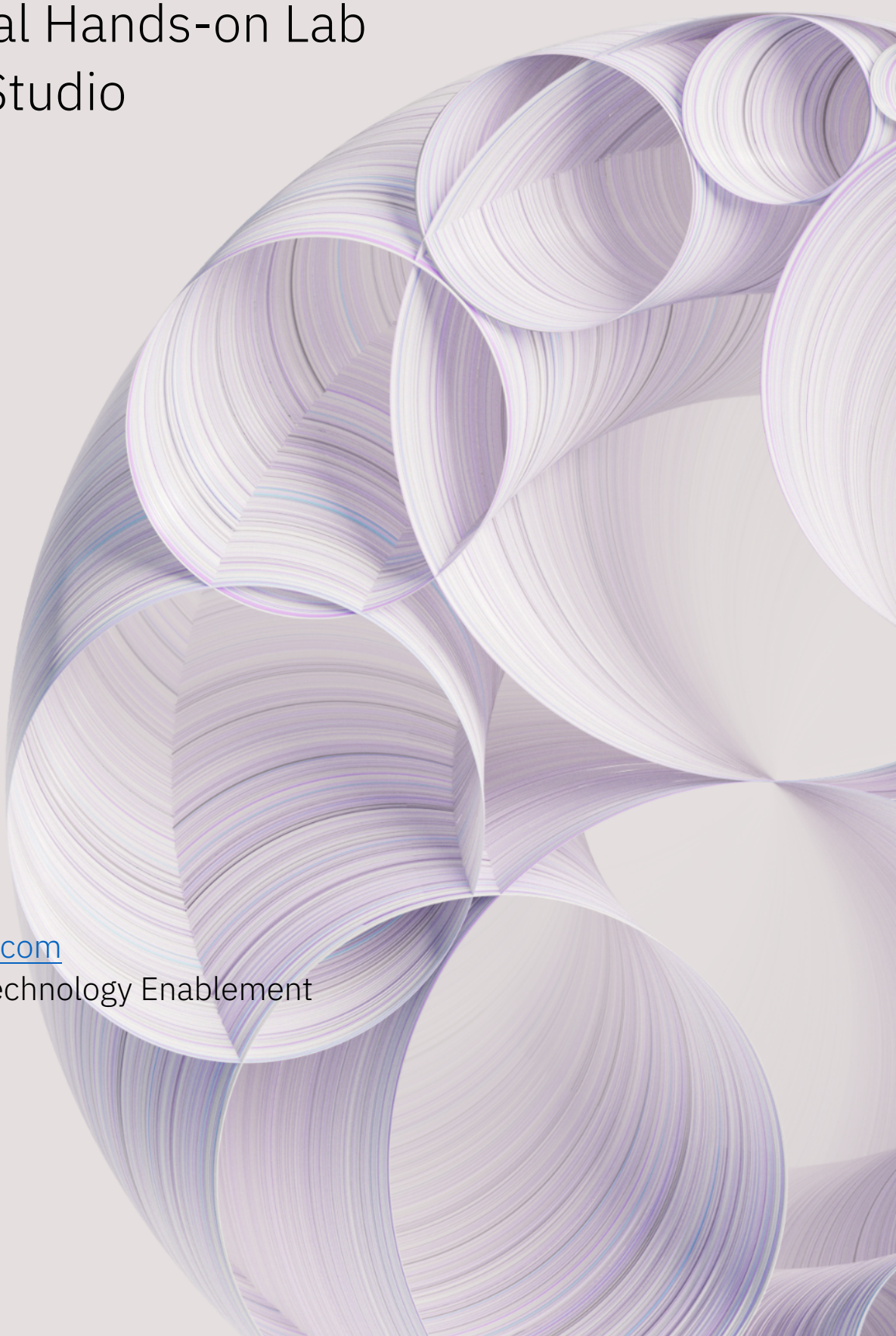
IBM watsonx.ai

Technical Hands-on Lab

Tuning Studio



Felix Lee
felix@ca.ibm.com
Worldwide Technology Enablement

Contents

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

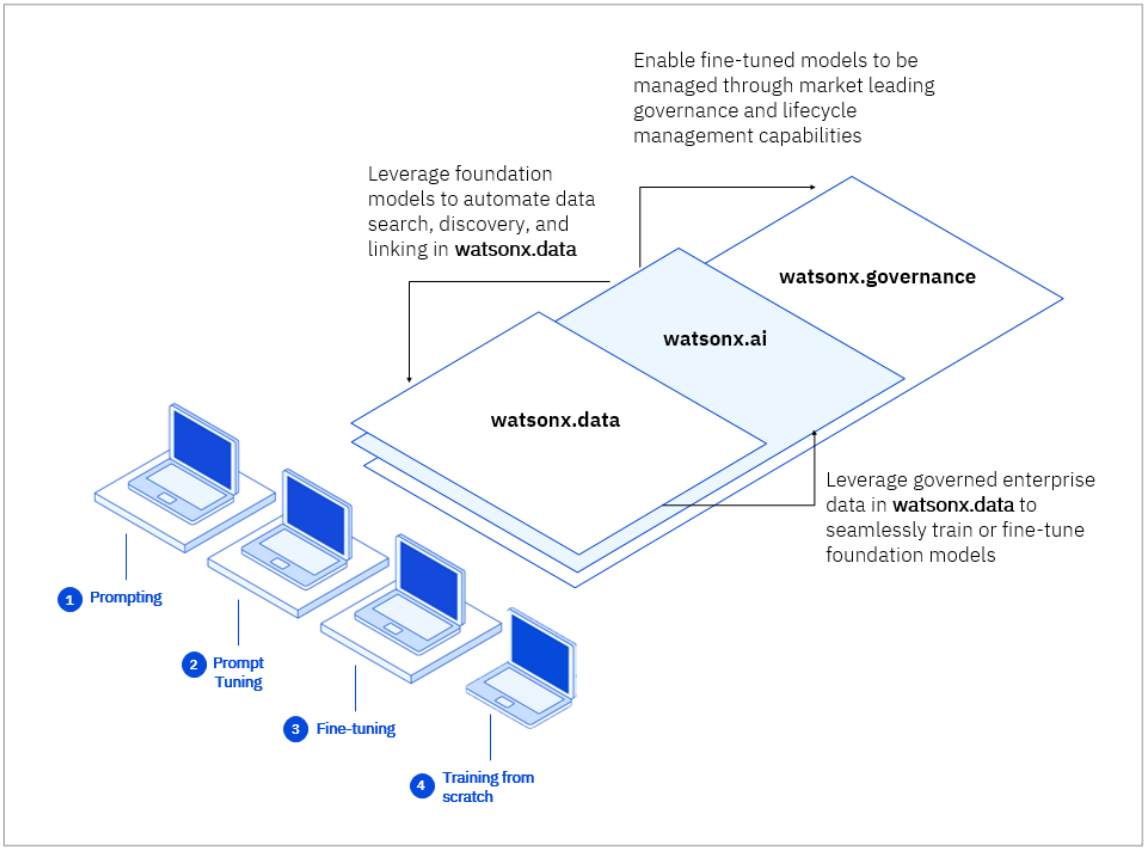


Figure 1: IBM watsonx platform

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 Part II is a supplement to the IBM watsonx.ai Technical Hands-on Lab Part 1. It is assumed that you have completed this first Hands-on lab. Detailed instructions and screen flow available there are not repeated here.

In this lab, you will be introduced to the following watsonx.ai capabilities:

- Explore differences between prompt engineering and prompt tuning
- Explore the Tuning Studio on the IBM watsonx.ai console
- Perform prompt tuning on a foundation model using a set of labeled data

You can use the same procedure from Part 1 of the lab to set up an environment to work with the labs here.

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 should not affect how the labs work, 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:

- 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.

- There is ongoing tuning of the models. If the models themselves are updated, then some of the results may vary.

Please post any questions on the #data-ai-demo-feedback Slack channel (IBMers 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 (IBMers 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 (IBMers) and #watsonx-ai-enablement Slack channels (IBMers 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 can use **IBM TechZone** to perform the exercises in this lab. The detailed setup instructions are available in this document (Section 4 of the L3 Lab Part 1). You must complete the steps before you can proceed with any of the exercises in this lab.

It is assumed that you have completed Part 1 of the L3 lab (or are familiar with the concepts and contents).

5. Prompt Tuning

Prompt tuning is accessible via the watsonx.ai Tuning Studio. Note that prompt tuning is different from prompt engineering.

In prompt engineering, a user is modifying what is fed into the foundation model. These are "hard prompts" modified by the user. This is useful in providing context, simple examples, and suggestions for output structures to the model. However, there are limits to how much prompt engineering (even with one-shot or multi-shot prompting) can do.

5.1 A classification use case

Consider the following classification use case. Suppose you are tasked with automating the triage of incoming complaints (a classification task). For your company, you need to route a complaint to all applicable departments among the following:

- Planning
- Development
- Service
- Warehouse
- Level 3 (L3 for short)
- Packaging
- Marketing

Some business rules exist for each department. Some of these include:

- Service/Support complaints go to **Service**.
- Skills issues that need support go to **Service** and **L3**
- Out-of-stock/missing items complaints go to **Warehouse**
- If the item is being discontinued, the complaints go to **Warehouse** and **Planning**
- Missing feature requests go to **Planning** and **Development**
- Complaints that are related to the perception of what the business does or does not provide go to **Marketing** and may involve **Planning** and **Development**.
- And more … based on your business process

This is a good generative AI use case as a large enterprise must handle lots of complaints and they must be routed properly to the correct department for rapid response. Such routing tasks need to be automated without spending a large number of human resources. Generative AI is also better at handling incoming complaints in natural language.

However, there are complexities with this use case. Large Language Models (LLMs) are very good at classifying with single labels (such as positive vs. negative sentiments) but will have a tougher time with business categories which may have very different meanings than the training data for the LLM.

In this case, sometimes the business requires multiple labels output, and there is specific business logic for why a particular output label is used. Prompt engineering, as you will discover, will have a tough time properly classifying all complaints.

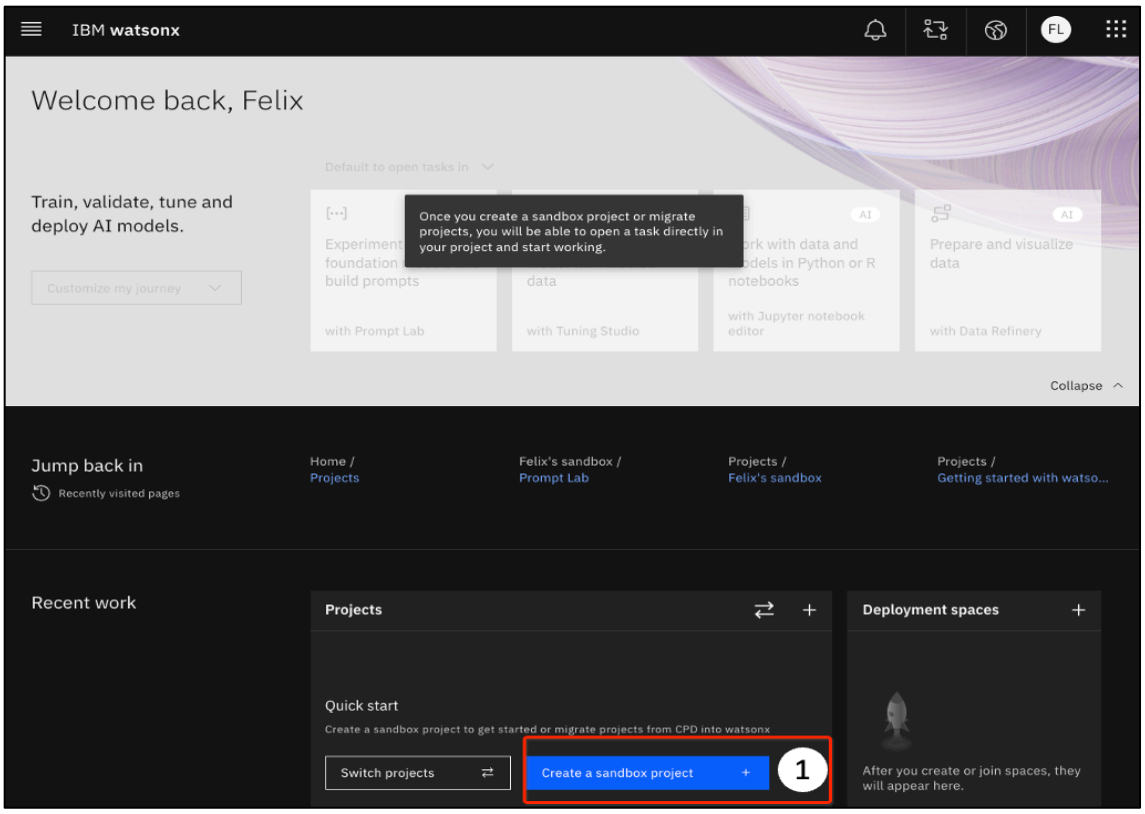
In prompt tuning, instead of human-generated hard prompts, a user provides a set of labeled data to tune the model. For this example, you will provide a list of sample complaints and the associated departments to be notified. Watsonx.ai will tune the model using this data and create a “soft prompt”. This does not change any of the existing model weights. Instead, when a user enters a prompt, it is augmented by the soft prompt and is passed to the model to generate output.

Important note: Not all models are available for prompt tuning right away. The **flan-t5-xl-3b** model (available December 2023) and the **llama-2-13b-chat** model (available February 2024) are currently available for tuning in watsonx.ai. Other models will be rolled out shortly. This lab is created using the **flan-t5-xl-3b** model and you should use it as well. Using **llama-2-13b-chat** or other models will exhibit different behaviors.

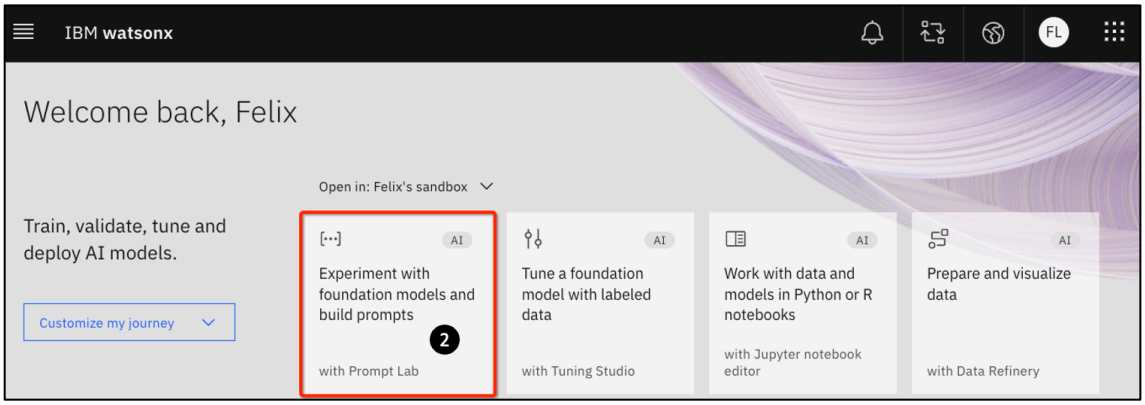
5.2 Using prompt engineering.

For this part of the lab, you will use the **flan-t5-xl-3b** model and see how much you can accomplish with prompt engineering.

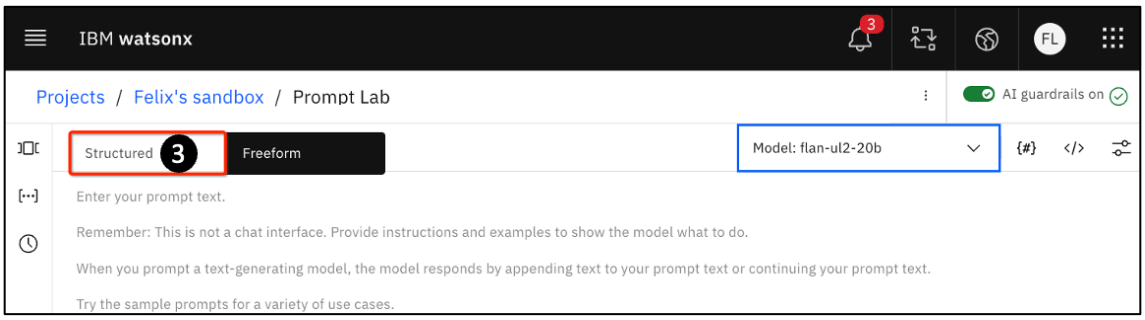
1. Open the watsonx.ai console. If the sandbox project has not been created, click the **Create a sandbox project** button to create it.



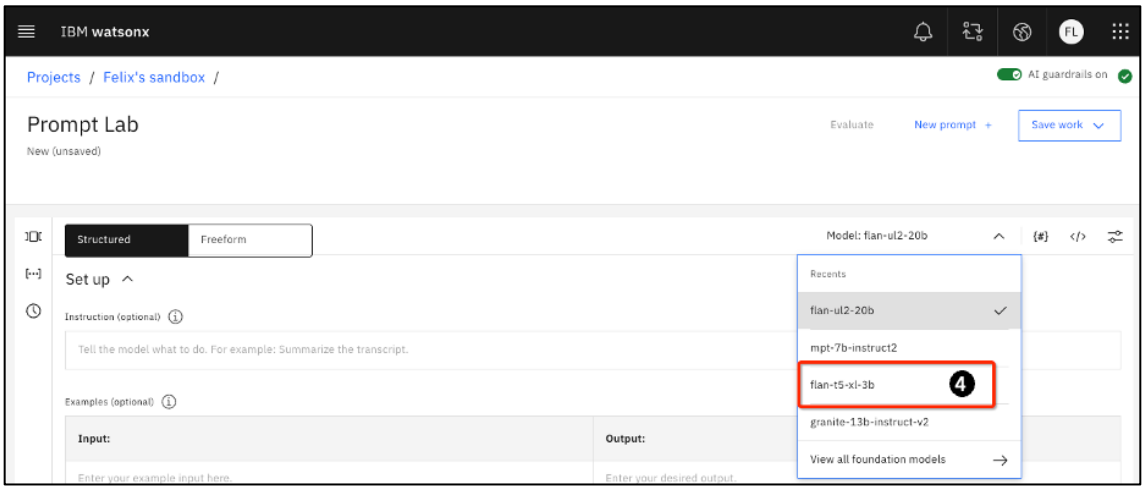
2. Select the **Experiment with foundation models and build prompts** tile.



3. The Prompt Lab pages open. It should be in the **Structured** mode. If not, select the **Structured** tab.



4. Click on the **Model** dropdown and select the **flan-t5-xl-3b** model (you may have to pick it out by clicking on **View all foundation models**).



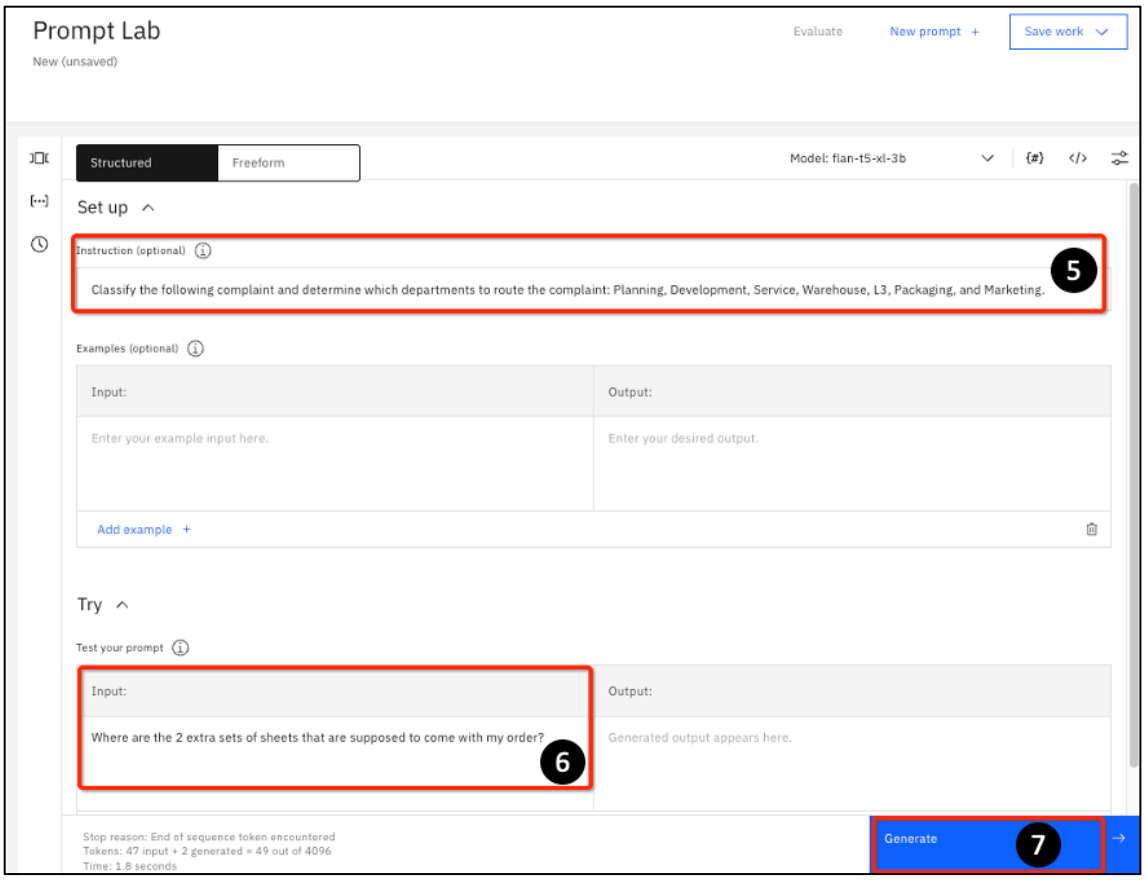
5. Enter the following text into the **Instruction** field:

```
Classify the following complaint and determine which departments to route the complaint: Planning, Development, Service, Warehouse, L3, Packaging, and Marketing.
```

6. Enter the following text into the **Input** field in the **Try** section:

```
Where are the 2 extra sets of sheets that are supposed to come with my order?
```

7. Click **Generate**.



8. The **flan-t5-xl-3b** model returns a completion of **Marketing**. This is not an unreasonable answer given that the flan model has no understanding of the business context. Since the complaint speaks of missing something "extra", the flan model gave its best shot to generate the **Marketing** completion.

   Given the business use case and the background information (in Section 7.1), you want the model to respond with a completion of **Warehouse** and **Packaging**.

   Note: Try this out. Click the **Generate** button several more times. You should get the same answer (as expected). You will return to this point later.

9. Click on the **Model parameters** (⚙) icon.

10. On the slide-out panel, you see the following settings:

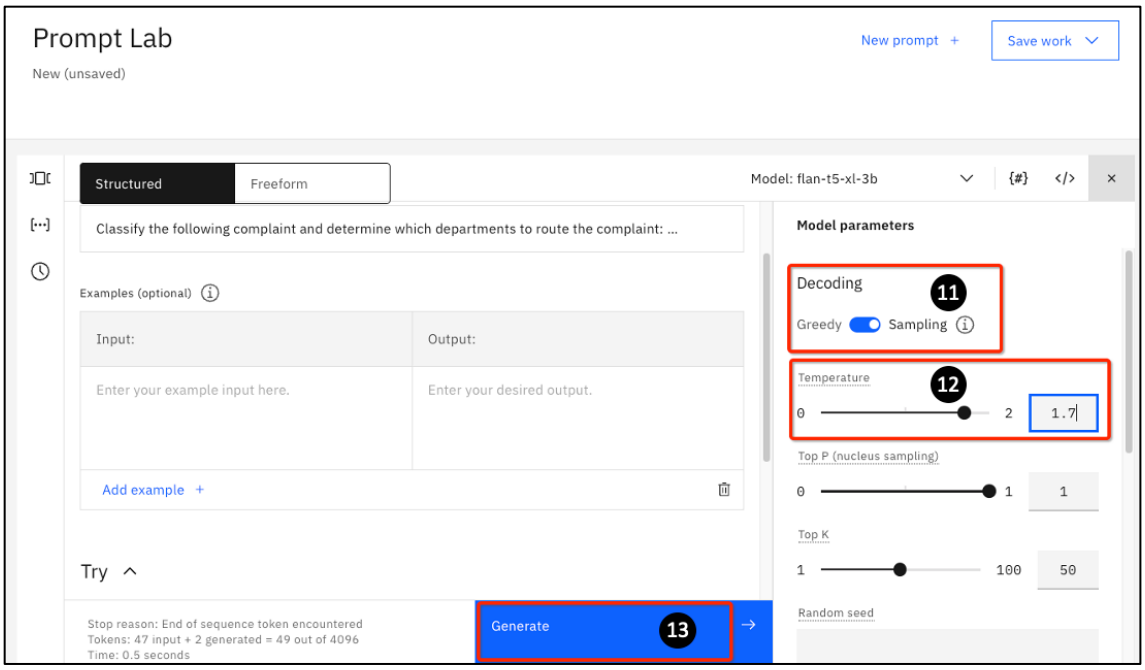
Decoding	Greedy
Repetition penalty	1
Stopping criteria	not set
Min tokens	0
Max tokens	20

11. Change **Decoding** from **Greedy** to **Sampling**. More parameter settings are visible:

Temperature	0.7
Top P	1
Top K	50
Random seed	not set

12. Change the **Temperature** setting from **0.7** to **1.7** to allow more variations.

13. Click **Generate**.

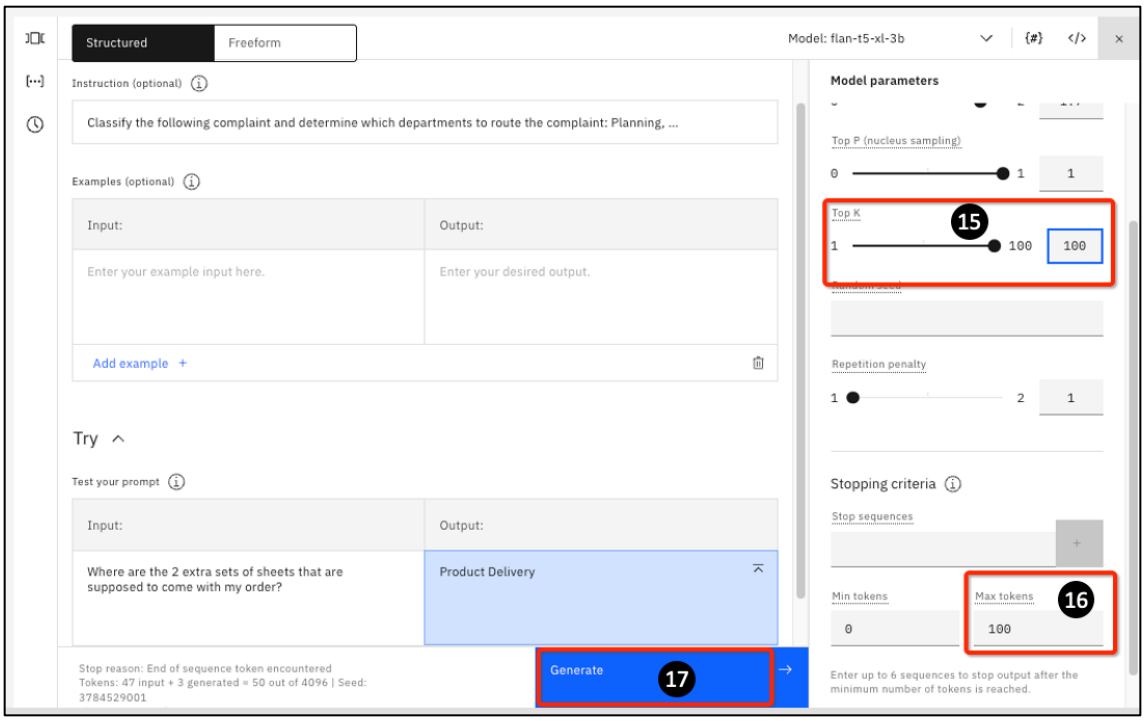


14. The **flan-t5-xl-3b** model returns with **Promotion** this time. This is again reasonable as this is a complaint about something missing so the problem can be with delivery. But this is not what you want, and **Promotion** is not a proper category.

15. Next, update **Top K** to **100** – this lets the model consider more possible outcomes.

16. Change **Max tokens** from **20** to **100**. This allows the model to be more verbose (or wordy) in its output. This tests whether this can prompt the model to output more than one department.

17. Click **Generate**.



The **flan-t5-xl-3b** model now provides **Marketing** as a completion.

Note: You *may* get a different output from **Marketing.** Regardless of what the initial completion was, now try repeatedly clicking **Generate**. You may see these:

- Marketing
- Warehouse
- Service
- Something else (not part of the categories you provided)

Why are you seeing this?

- Recall earlier in Step 8, with repeated clicking of the **Generate** button, you get the same answer. At Step 8, you were using the **Greedy Decoding** mode. The model is always going for the highest possibility according to its internal algorithm.

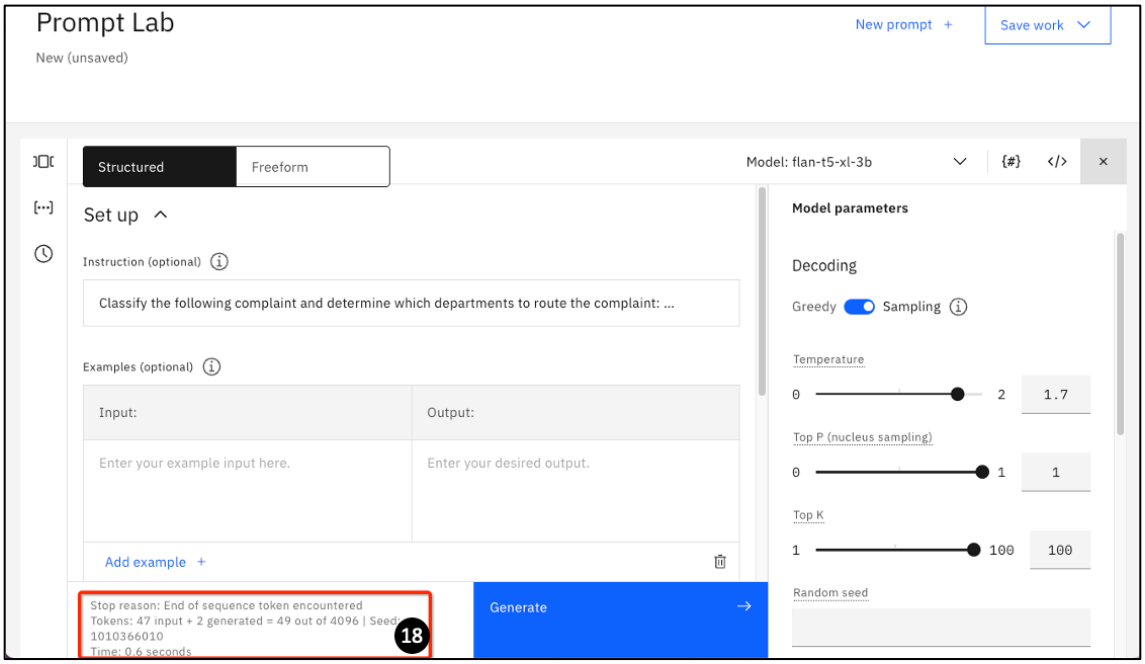
- Here, however, you are using **Sampling Decoding** with a large **Temperature** setting and the maximum value of **Top K**. This introduces high variability, and the model is considering all possibilities for completion. This is why you keep seeing different results.

- You can specify a value (does not matter what it is) for **Random seed**. That will allow the model to choose from a wider range of possibilities the first time. But after that, repeatedly clicking **Generate** will yield the same output.

18. Scroll to the bottom of the page and note the information regarding the current completion.

    For now, note the following (you will use this information later):

    Tokens: 47 input + 2 generated = 49 out of 4096



Section summary

- Large Language Models (LLMs) can provide reasonable outcomes on business-specific use cases such as classification.
- However, out-of-the-box LLMs are generally not good enough where there are specific business languages and operational details, especially where terminologies and business requirements are constantly being updated.
- In these classification use cases with specific business requirements, varying inference parameters are not useful. The business is not looking for creativity, but precise identification.
- When you use Sampling Decoding, you can introduce variability. This can be an advantage for certain use cases but not for this classification use case. Be aware that this can also introduce different results from one user to another.

You will now try one-shot and multi-shot prompting.

5.3 Using one-shot and multi-shot prompting

Since model parameters and simple prompt instructions do not seem to work, you will now try one-shot and multi-shot prompting.

1. First, do a one-shot prompting by providing an example that outputs 2 items in the completion. This is to teach the model an example of what the business expects (matching the complaint to potentially more than one department).

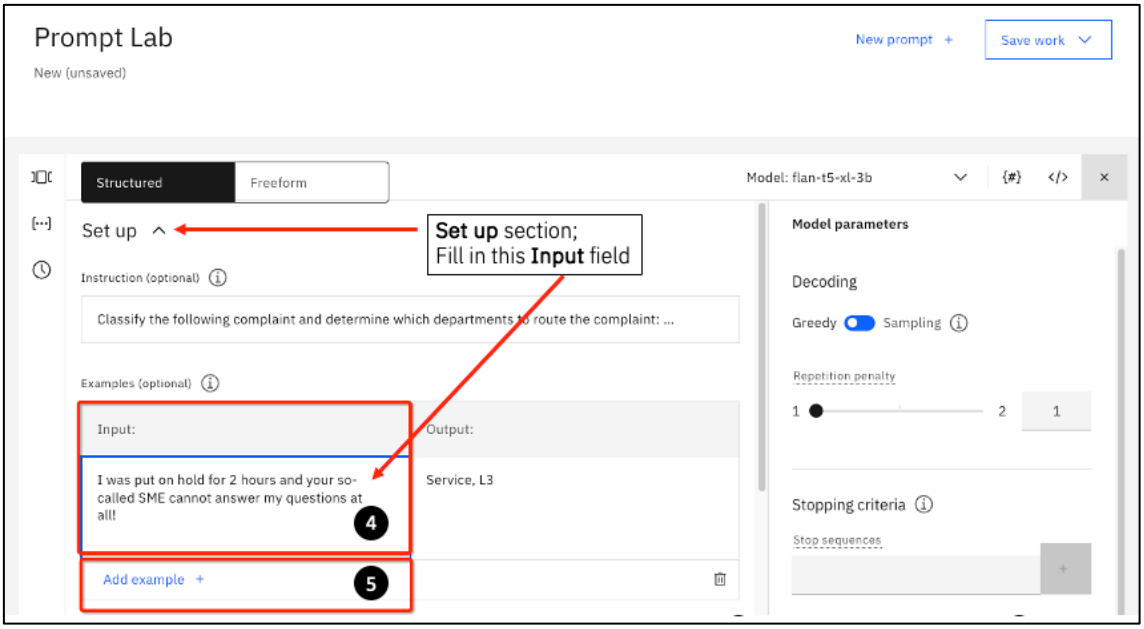
2. Ensure you are using the **flan-t5-xl-3b** model.

3. Reset **Decoding** to **Greedy**. You can leave **Max tokens** to 100.

4. Paste the following test in the **Input** field of the **Set up** section.

```
I was put on hold for 2 hours and your so-called SME cannot answer my questions!
```

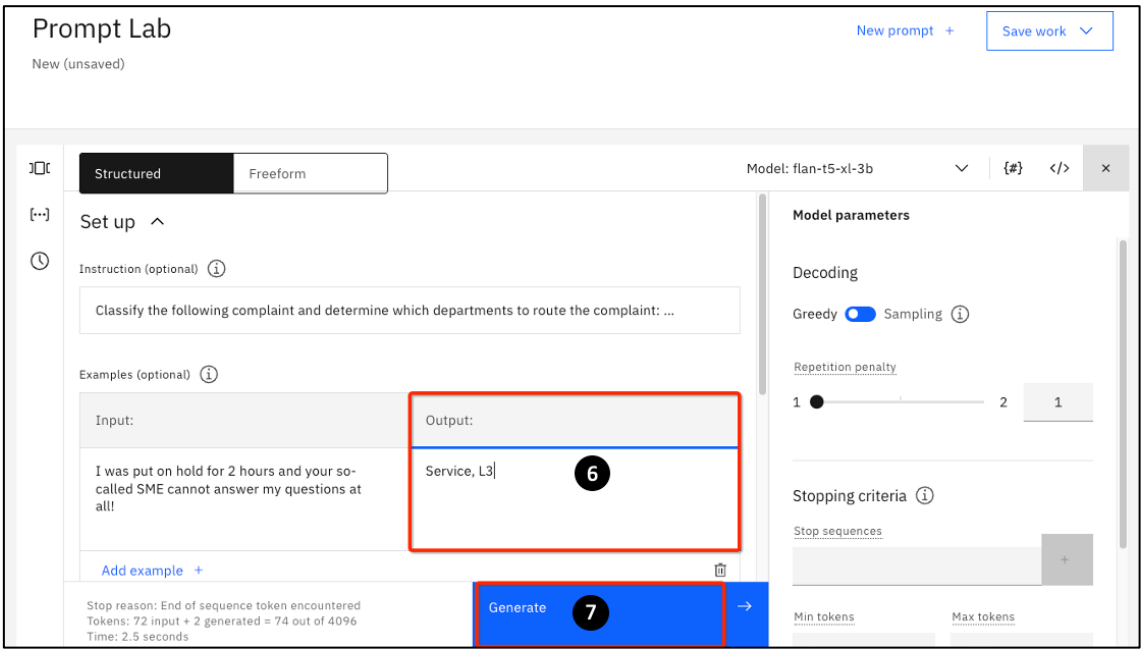
5. If you do NOT see an available **Input** line, simply click on **Add example +** to add a line.



6. Paste the following text into the corresponding **Output** field.

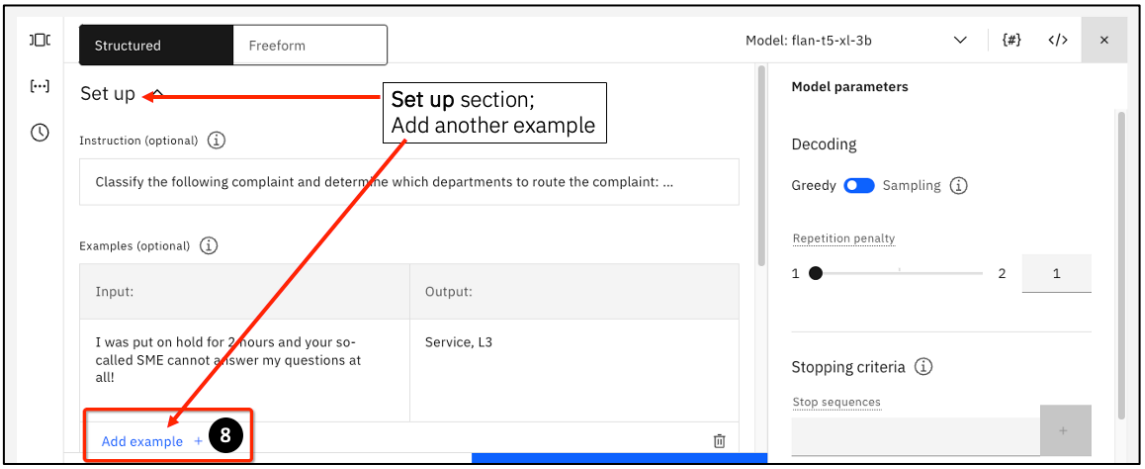
```
Service, L3
```

7. Click **Generate**.



The output is now **Service**. The one-shot prompting has changed but has not improved the completion.

8. Now you will use a 2-shot prompting. Click on **Add example +** from the **Set up** section to add another input line.



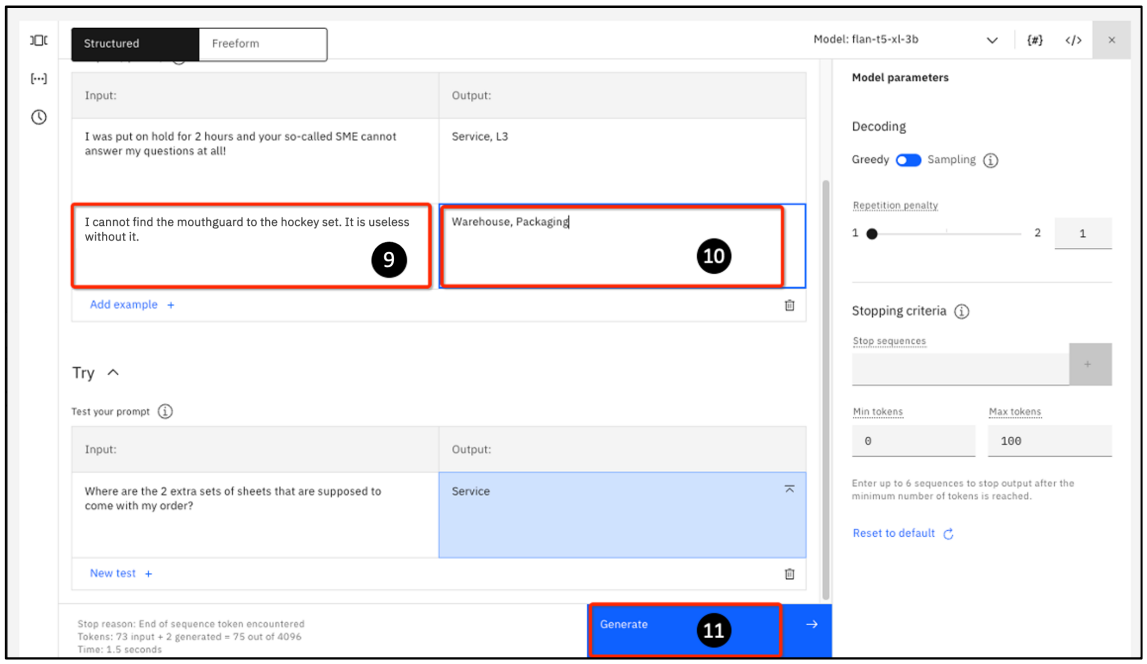
9. Paste the following into the newly created **Input** field area.

```
I cannot find the mouthguard to the hockey set. It is useless without it.
```

10. Paste the following into the **Output** field area.

```
Warehouse, Packaging
```

11. Click **Generate**.



12. The **flan-t5-xl-3b** model generates a completion of **Warehouse** this time. There is some improvement over the one-shot prompt in that the LLM identifies **Warehouse** this time.

13. Click on **Add example +** in the **Set up** section to add another input line (see Step 8).

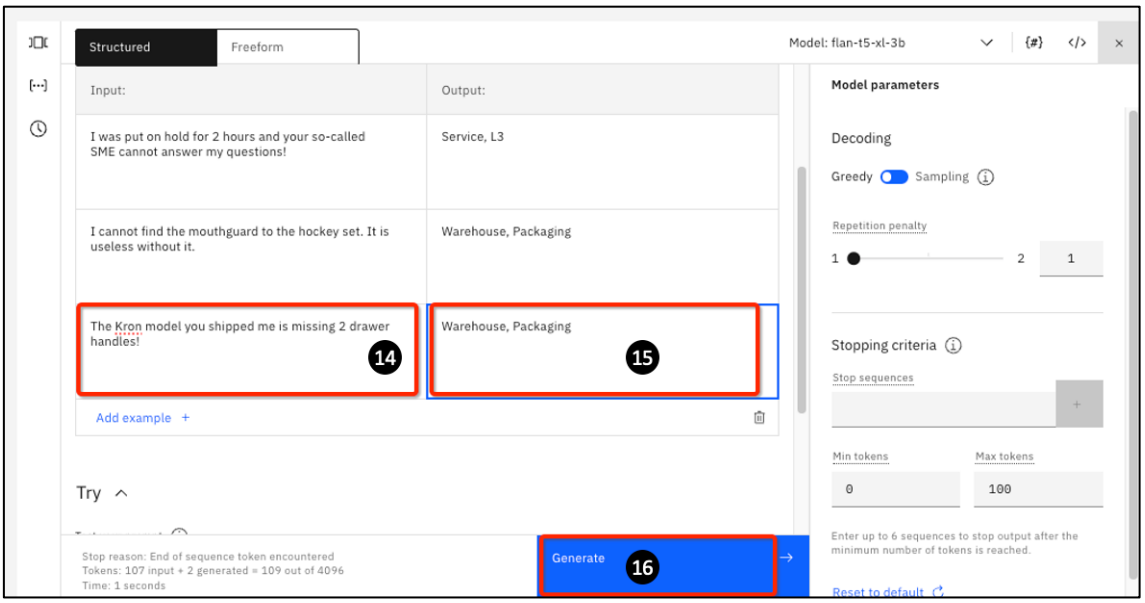
14. Paste the following into the newly created **Input** field area.

```
The Kron model you shipped me is missing 2 drawer handles!
```

15. Paste the following into the **Output** field area.

```
Warehouse, Packaging
```

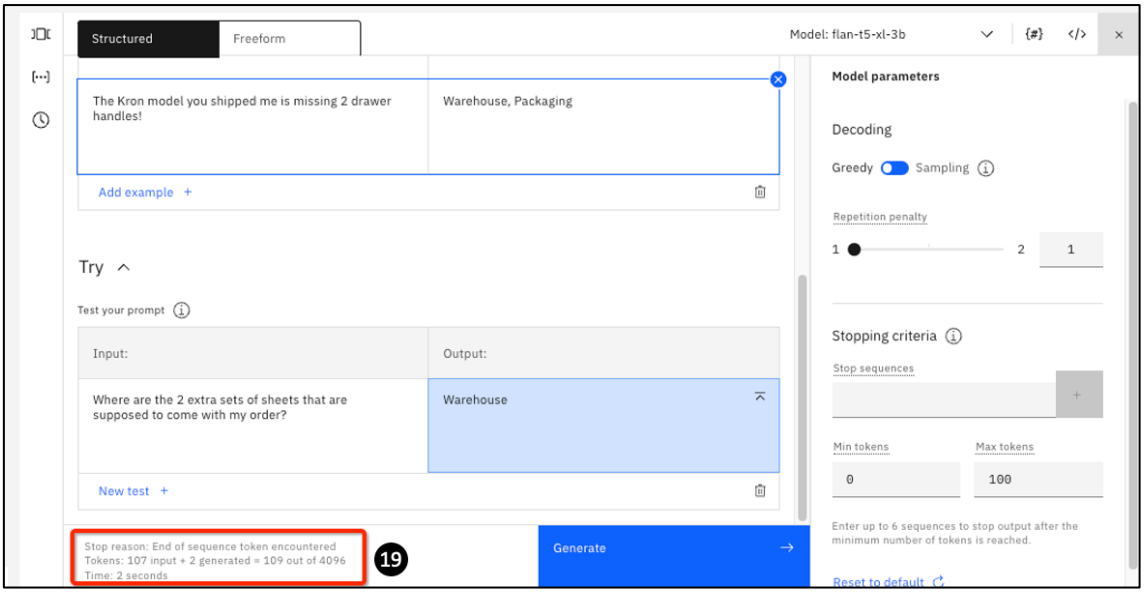
16. Click **Generate**.



17. The model has not improved its completion and still just provides **Warehouse** as the completion. The model has not learned to identify more than one recipient despite the 3-shot prompt.

18. You can try additional shots. Given enough examples that look very similar to the **Input** and **Output** in the **Try** section, you *can* get the **flan-t5-xl-3b** to repeat **Warehouse** and **Packaging** as the completion. But that just means the LLM can recognize *this* pattern. It is not likely to work on other patterns.

19. Before moving on to the next section, scroll down the page and once again capture the token usage:



Note: In this case, there are 109 input tokens, much bigger than before. For now, just note this information. You will use this information later in this lab.

Section summary:

- In this section, you tried using multi-shot prompting to help the flan-t5-xl-3b model understand how to perform a proper completion for your use case.
- However, while multi-shot can provide good results, it does not seem to learn that multiple targets for notification are allowed and desirable. It was still incorrect, despite every example passed in having multiple departments in the output.
- Compare tokens used in Section 7.2 and 7.3.
    - Input token count for Section 7.2 (base prompt engineering): 47
    - Input token count for Section 7.3 (with 3-shot prompting): 109

  There is no surprise here; 3-shot prompting requires more input. Keep these numbers in mind when you move on to prompt tuning.

5.3.1 A look behind the scenes

A Large Language Model (LLM) is a probability machine. Its knowledge comes from the data used to train it. The larger the data set, the more the LLM "knows". An LLM calculates the output based on the input prompt and the LLM's knowledge base. The more the LLM knows, the more options (or the larger the repertoire) the model can draw on for the completion.

By contrast, if the LLM does NOT know about a subject, it has nothing to draw on. The LLM might try to extract context from the prompt, or it might make things up (hallucinate), or in rare cases, the LLM might not provide a completion.

For example, if a model's training data does not include any information on automobiles, it cannot generate a sensible paragraph describing what a McLaren is, or the difference between a Ford 150 and a Ram 1500.

A more obvious example is a language one. Most LLMs can provide a fairly good completion with the prompt: "Write a paragraph describing Canada". However, unless the model has been exposed to other languages it would not know how to respond to the same question in another language (even if you simply replace the country name of Canada with its equivalent in another language).

The classification task you started looking at in Section 5.3 is another example of this LLM issue. The LLM you are using (flan-t5-xl-3b) would not have been built with information from the specific business-related data. It is not a human reading a complaint and automatically understanding how to route the complaint to multiple destinations. Of course, one can consider a human brain a huge LLM and it has a good understanding of business rules created by another human brain.

It is clear why multi-shot prompting does not work well:

- Complaints are filed in natural language and there are so many ways someone can say something – including being sarcastic, in some short form, or in poorly structured phrases.
- There can be many rules and combinations that even a 3-shot, 4-shot, or teens-shot prompting would not be able to adequately cover.

Next, you will use prompt tuning to improve the performance.

6. Prompt Tuning

You will need to download the Call center complaints JSONL file. Keep it somewhere local that you can use later.

Note: the file might be downloaded with an uppercase extension (.JSONL). If so, you need to change it to lowercase (i.e. change the name from **Call center complaints.JSONL** to **Call center complaints.jsonl**).

6.1 A look at the content of the JSONL file

In prompt tuning, you provide a set of labeled data to tune the model. In essence, the labeled data helps the model to understand the business data and requirements. The following is the partial content of the file.

{"input":"There is no instruction on how to assemble the foundation.","output":"Packaging"}
{"input":"The product is not working out-of-box.","output":"Development"}
{"input":"Your product broke after 2 weeks of usage!","output":"Service, Development"}
{"input":"This product does not work as advertised!","output":"Marketing, Development"}
{"input":"Some of the parts are scratched up even though the package looks to be new.","output":"Packaging"}
{"input":"This does not look anything like what you have on your website.","output":"Planning, Marketing"}
{"input":"The resolution of your projector is so bad, totally useless.","output":"Planning, Development"}
{"input":"This is so flimsy! It can't support my weight.","output":"Planning, Development"}
{"input":"My laundry still stinks after washing with your product.","output":"Development"}
{"input":"The LED display is so faint, I can't read it at all.","output":"Planning, Development"}
{"input":"I can't insert the memory card, it just won't hold it.","output":"Planning, Development"}
{"input":"There are multiple missing parts from the package.","output":"Warehouse, Packaging"}
{"input":"It is much shorter than I saw from your commercials.","output":"Planning, Marketing"}

The structure of the labeled data is clear. It is easier to see in an equivalent JSON format:

```
{
    “input”: There is no instruction on how to assemble the foundation.",
    "output":"Packaging"
}
```

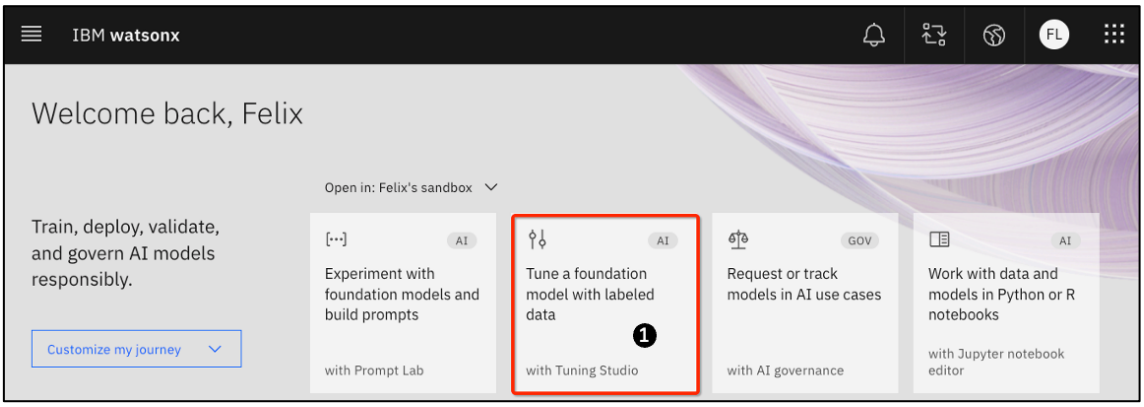
The **input** is a complaint, and the **output** is where the complaint should be routed. This file comes from the business, and it provides examples of complaints in natural language and which departments the complaints should be routed to.

In prompt tuning, this set of data is turned into a soft prompt and is used to enhance the runtime hard prompt the user provides. The list of examples teaches 2 things:

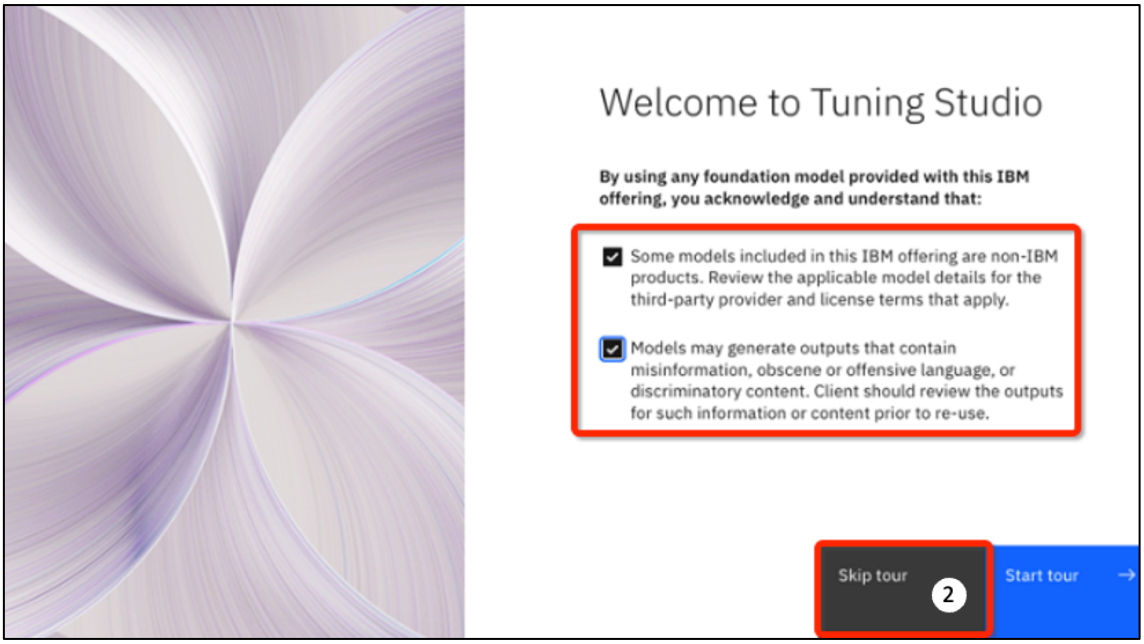
1. Where specific complaints are being routed to for this business.
2. The business allows routing a complaint to multiple destinations.

6.2 Prompt tune flan-t5-xl-3b with labeled data

1. Log in to the watsonx.ai console and select the **Tune a foundation model with labeled data** tile.



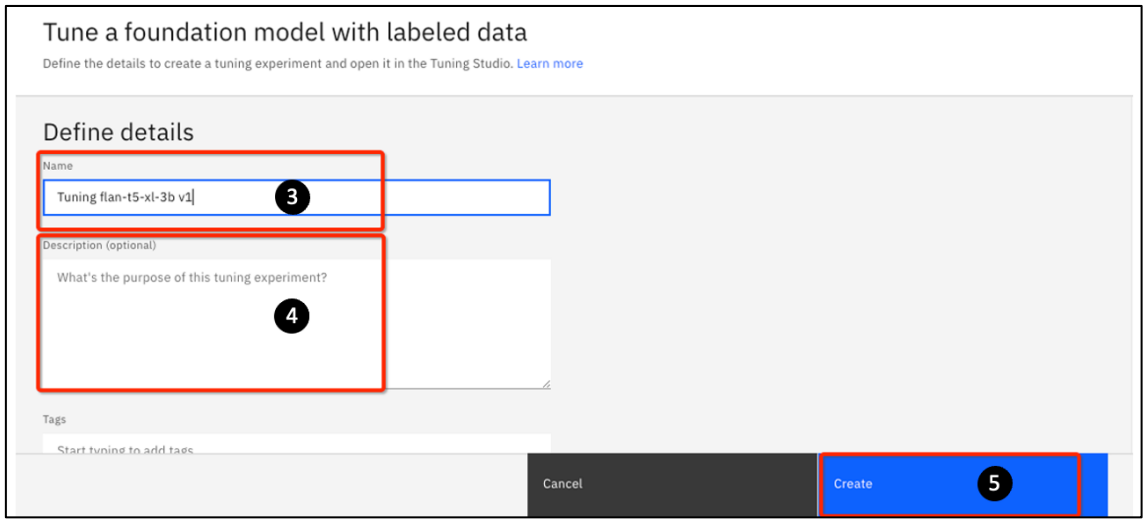
2. If you have never logged in to the **Tuning Studio**, you will see the following, you can simply agree to the conditions and click **Skip tour**



3. The **Tune a foundation model with labeled data** page opens. Provide a **Name** for the tuning experiment, such as **Tuning flan-t5-xl-3b v1**.

4. Provide an optional **Description** if you want.

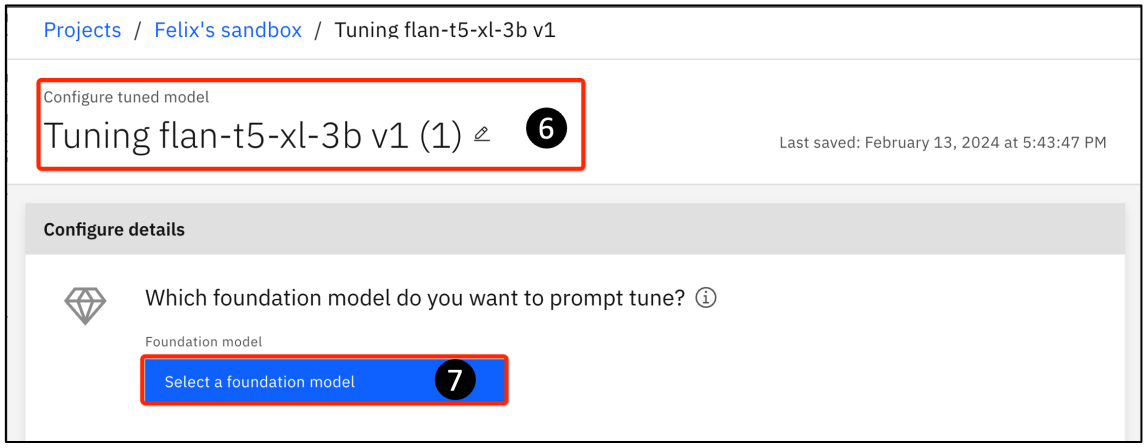
5. Click **Create**.



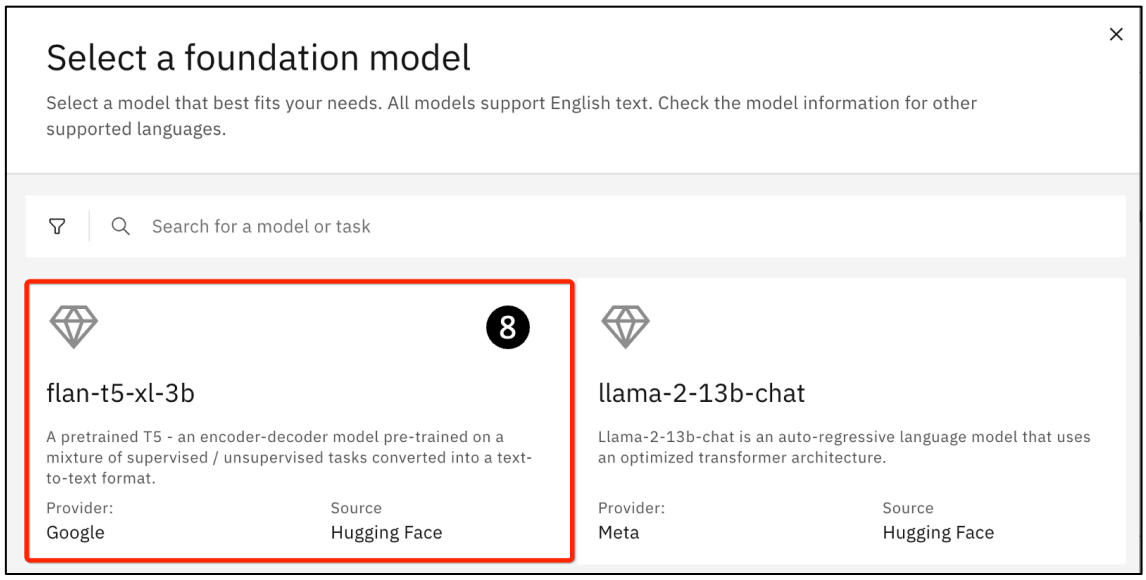
6. The **Configure tuned model** page opens. Watsonx.ai will add a versioning **(1)** to the name, allowing you to use the same set of data with different models or configuration parameters. You can change the name or remove the versioning if you want. In this lab leave the name as is.

7. Not all models are available for prompt tuning right away. The **flan-t5-xl-3b** model and the **llama-2-13b-chat** model are currently available (as of February 2024) for tuning in watsonx.ai. Other models will be rolled out shortly. This lab is created using the **flan-t5-xl-3b** model and you should use it as well. Using **llama-2-13b-chat** or other models will exhibit different behaviors.

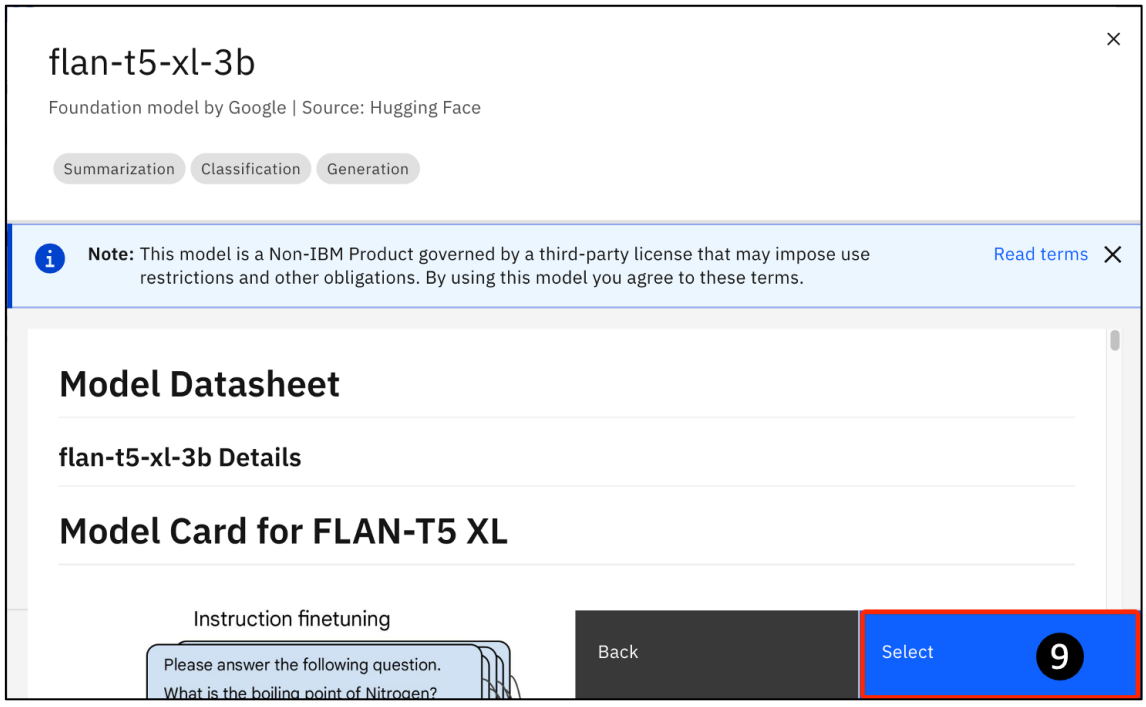
   Click on **Select a foundation model**.



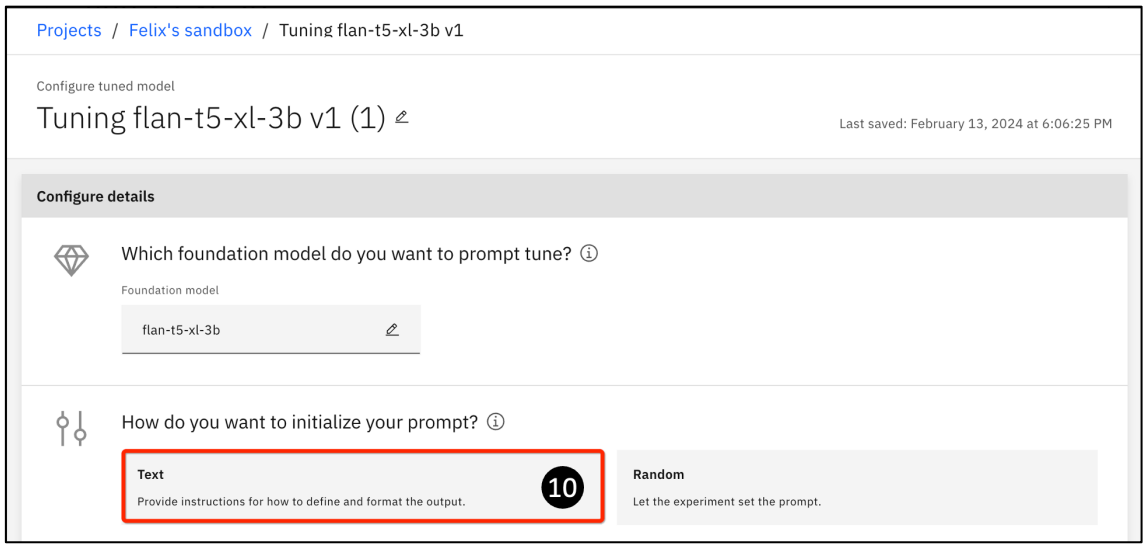
8. Select the **flan-t5-xl-3b** model.



9. The information page on the **flan-t5-xl-3b** opens. Click on **Select**.



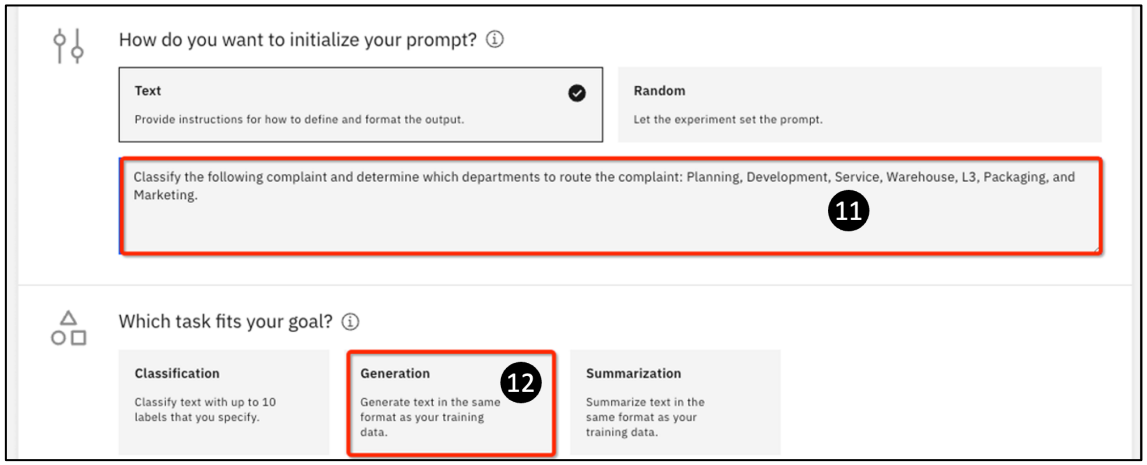
10. You are returned to the **Configure tune model** page. In the **How do you want to initialize your prompt?** section, select the **Text** tile.



11. You are asked to **Enter a task description and instructions**. Paste in the following text:

> Classify the following complaint and determine which departments to route the complaint: Planning, Development, Service, Warehouse, L3, Packaging, and Marketing.

12. You are asked **Which task fits your goal?** Select **Generation**.



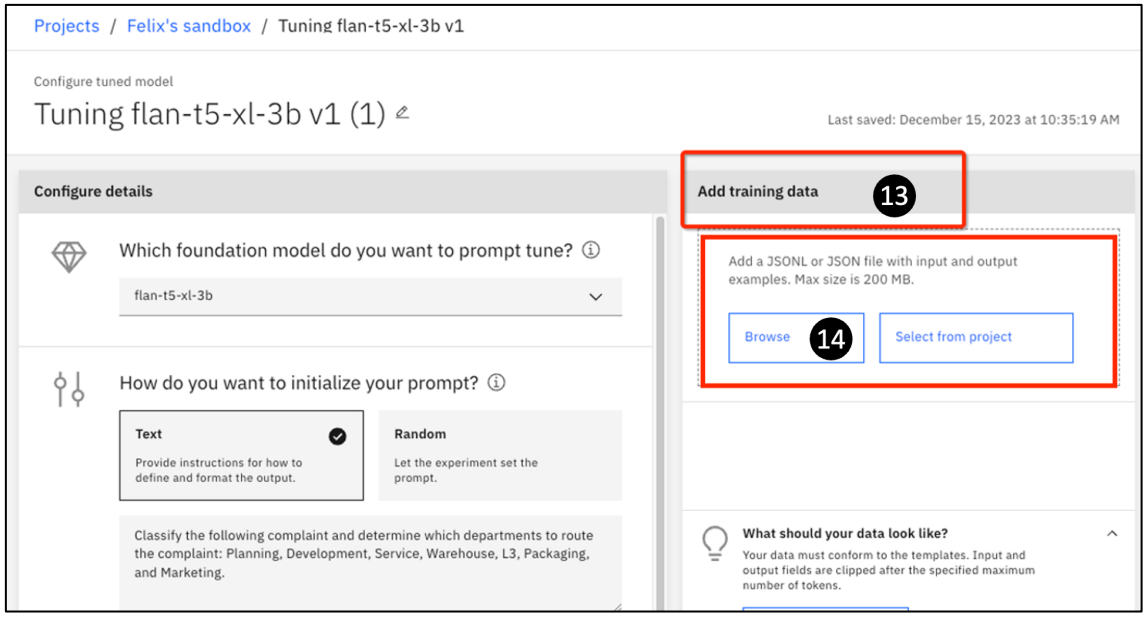
Note: Why **Generation**? This looks more like a **Classification**, right?

This use case may seem to fit more naturally into a **Classification** use case. At the moment (Feb 2024) the **Classification** use case supports Single-label classification with up to 10 classes. This use case, however, needs the LLM to output completion with multiple classifications (for example, **Warehouse** and **Packaging**). For now, **Classification** is not the proper goal to use.

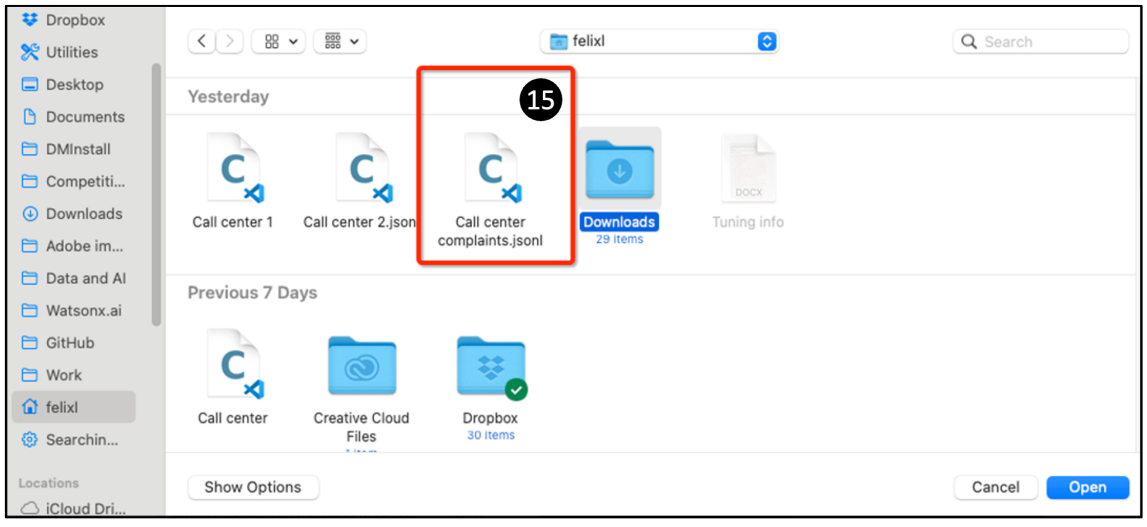
Generate, on the other hand, can generate text in a certain style and format. You will use the labeled data set to train the model to output the proper completion.

13. The **Add training data** column appears.

14. You can use the **Browse** button to add the Call center complaints.jsonl file (which you should have downloaded already, or use this link to download it now). If the file you downloaded has a file extension of **.JSONL**, rename it to use lowercase **.jsonl**.



15. Select the **Call center complaints.jsonl** file.



Watsonx.ai will perform a quick verification check on the file. If there is any error message, you will need to fix the JSONL file and re-load the file.

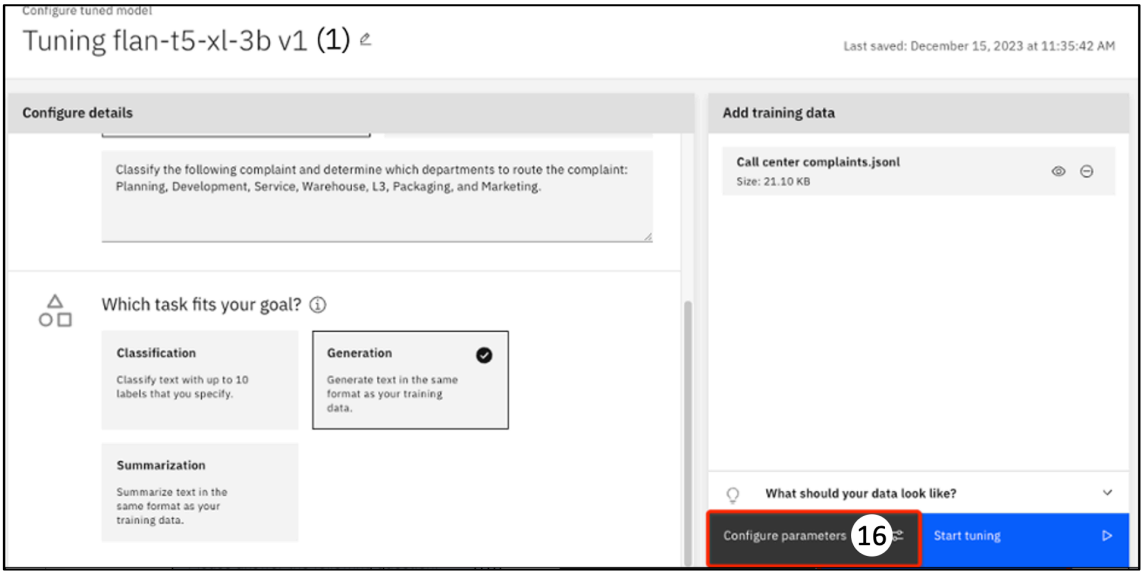
A proper JSONL file for prompt tuning has lines with the following format:

{“input”:“<input text>”,“output”:“<output text>”}

For example:

{“input”:“ I ordered 5 units but you only shipped me 2!”,“output”:“Warehouse, Packaging”}

16. Click **Configuration parameters**.



17. The **Configure parameters** page opens. For now, there is no need to modify the parameters but here is some information on the parameters:

    - **Batch size** – This is the number of samples to work through at one time. The range is between 1 and 16 (the default). Generally, the smaller the training data set (in this case the number of entries in the **Call center complaints.jsonl** file), the lower the **Batch size** can be.

      However, keep in mind that if the batch size is too small compared to the number of samples in the labeled data set, the longer (and more costly) the tuning.

    - **Number of epochs** – The number of times to cycle through the training data set. The range is between 1 and 50 (20 is the default). The higher the number, the longer it takes to complete the tuning. There are a couple of factors in play:
      - A higher number of epochs means it is more costly (you are using more resources).

    - A higher number of epochs usually means the model is improving in prediction. However, there is a levelling effect where further recycling of the training data will not bring meaningful improvement.

- **Learning rate** - This determines how fast the neural network will progress towards the optimal "learn" state (when prediction is closest to reality). If the rate is too low, the model is likely to pick up a lot more nuances but will take a very long time to get to the optimal state. If the rate is too fast, the model may miss the actual optimal point. The range is between 0.01 to 0.5 (0.3 is the default). See Appendix A for more details.

- **Accumulate steps** – This is the number of training steps (or batches) you want to accumulate before updating the internal parameters of the model. The range is between 1 and 128 (16 is the default).

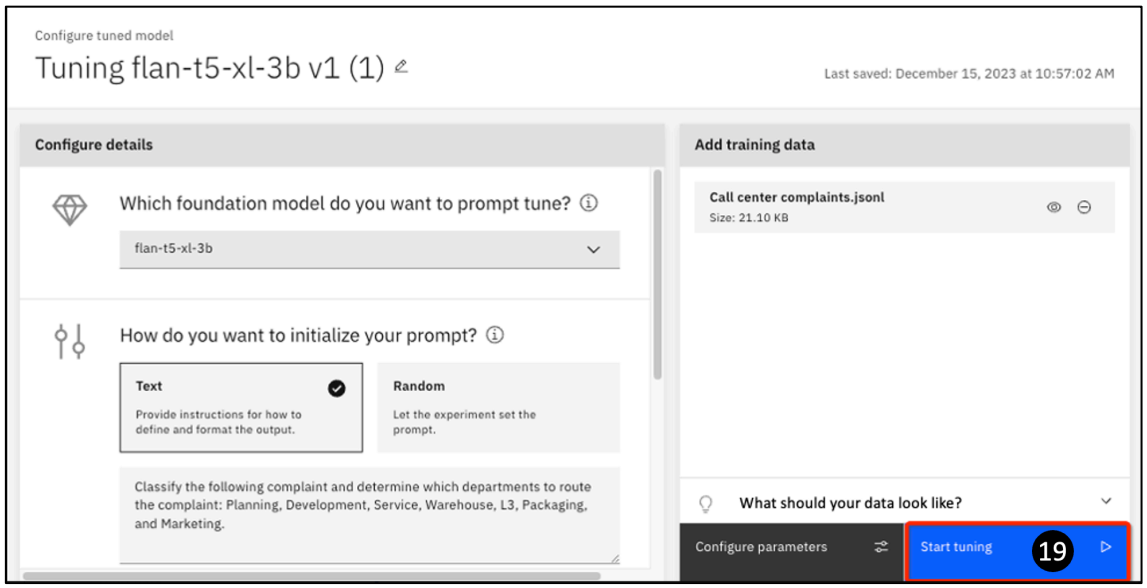
  For example, if **Batch size** is 10 and **Accumulate steps** is 10, then watsonx.ai will process 10 examples from the labeled data set each time, and after 10 such batches, the internal parameters will be updated.

A detailed explanation of these parameters is out of the scope of this guide. More will be said later on the **Number of epochs**.

For now, simply leave everything at its default values.

18. Click **Cancel** to exit this page.

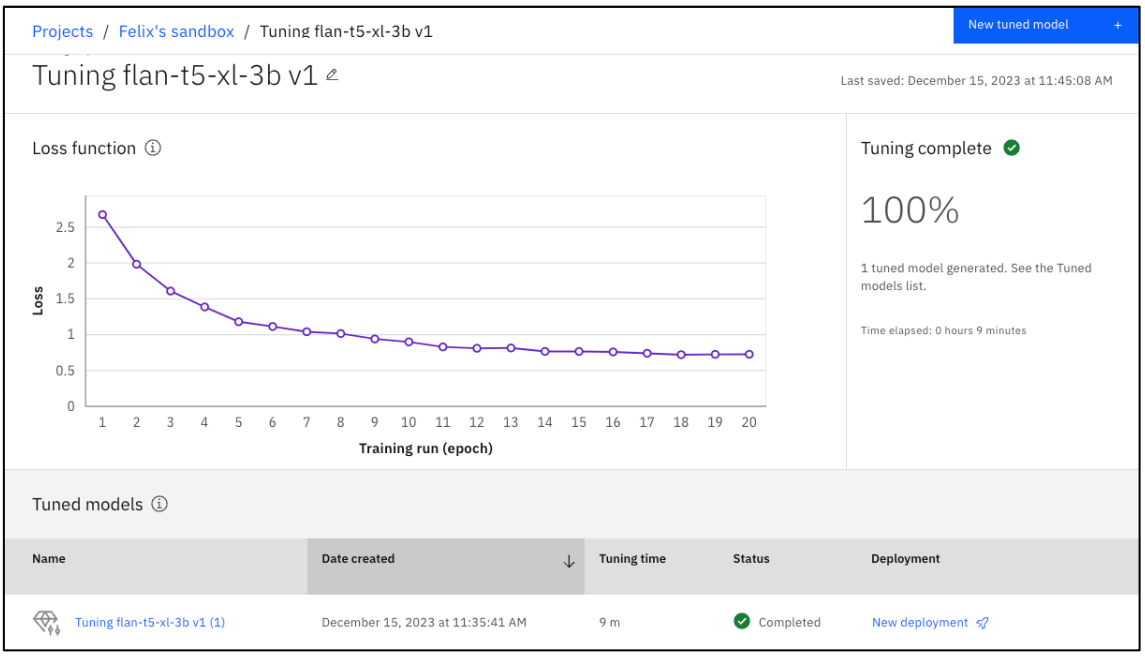
19. Click **Start tuning**.



20. You are returned to the **Tuning experiment.** Note that this can take quite some time (between 5-10 minutes so be patient).

Note: this step provides a view into the cost of tuning. This is only prompt tuning and not fine-tuning (where you are changing the actual model). Fine-tuning will take much longer.

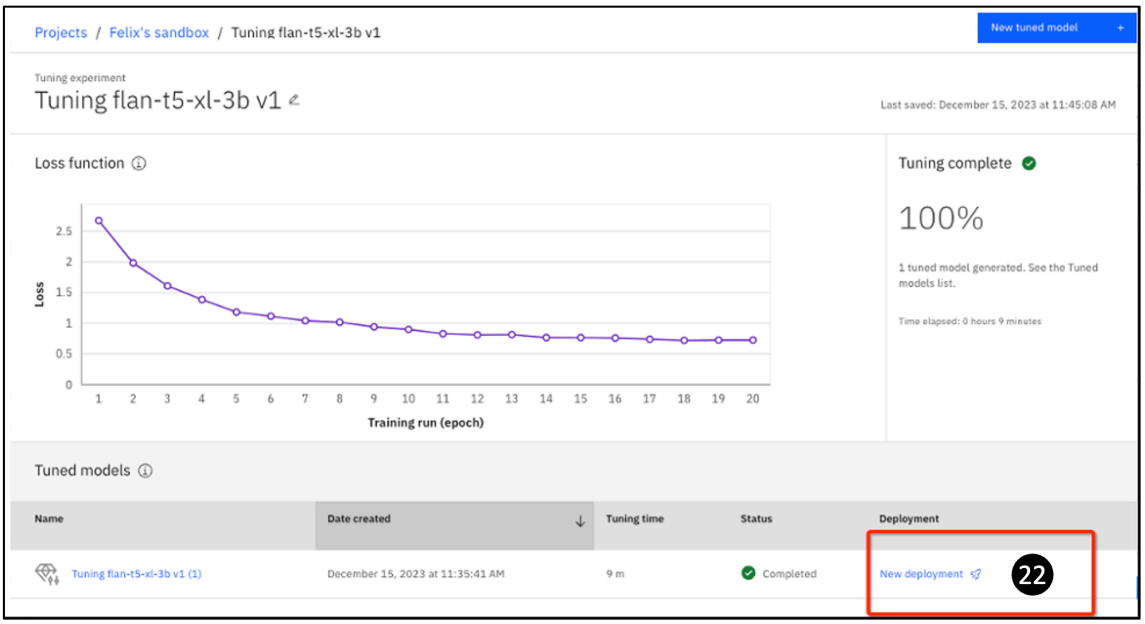
21. The tuning process completes, and you will see the **Loss function** plotted.



Note: At a high level, the **Loss function** measures the difference between what the LLM predicts and what is the actual result. There are several things to observe:

- When tuning began, the value of the **Loss function** value was high. The LLM's completion is quite different from the expected output.
- As expected, with every re-training (epoch), the **Loss function**'s value came down as the model became better at predicting.
- You see the levelling effect started around epoch 18. The model is no longer gaining a lot more knowledge via additional epochs. So, while a higher **Number of epochs** means a more accurate model in general, you can very quickly reach a point of diminishing return. Remember that the more epochs you run, the more costly it is.
- The **Loss function** levels off around 0.7. This likely means that the data set can be improved. However, this is sufficient for the current lab.

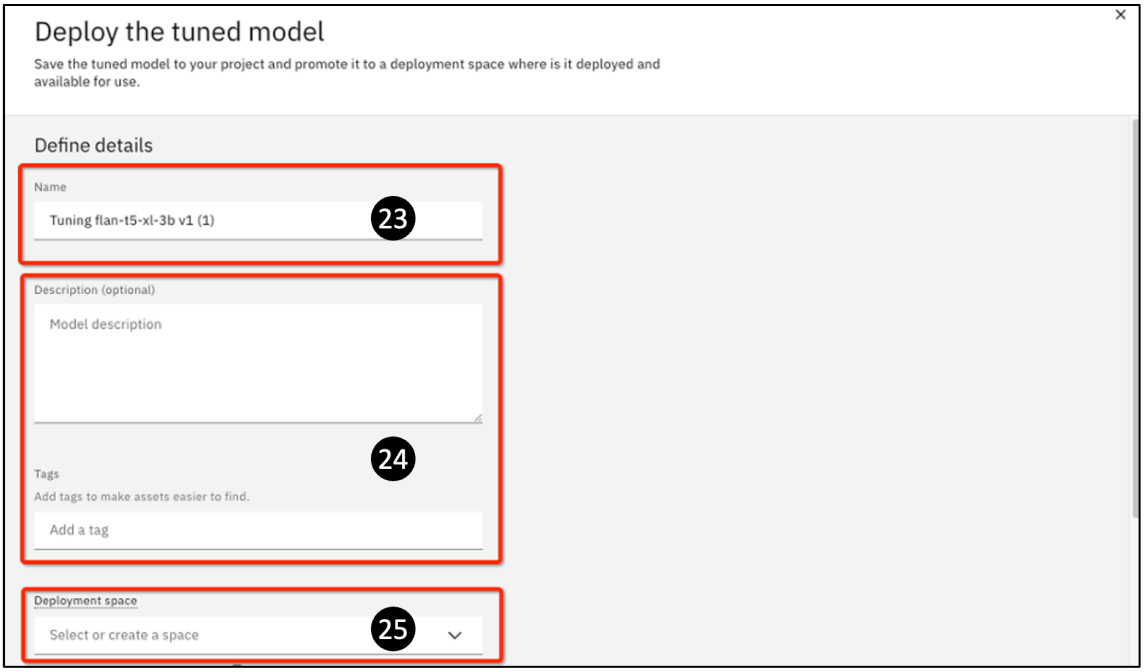
22. The tuned model needs to be deployed before it can be used. Scroll down on the **Tuning experiment** page and click on **New deployment**.



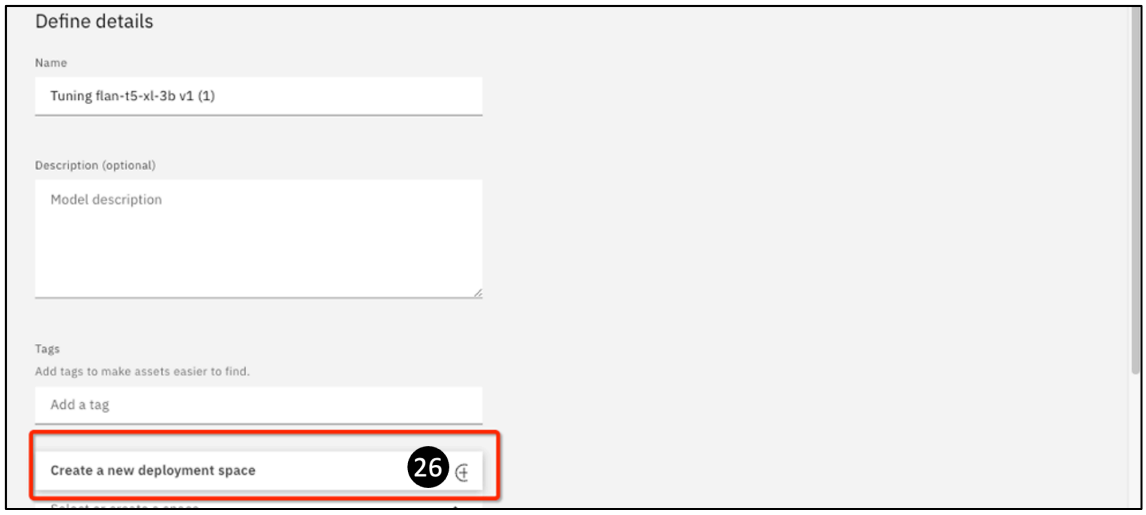
23. The **Deploy the tuned model** page opens. Notice that the **Name** is **Tuning flan-t5-xl-3b v1 (1)**.

24. You can add an optional **Description** or **Tags**.

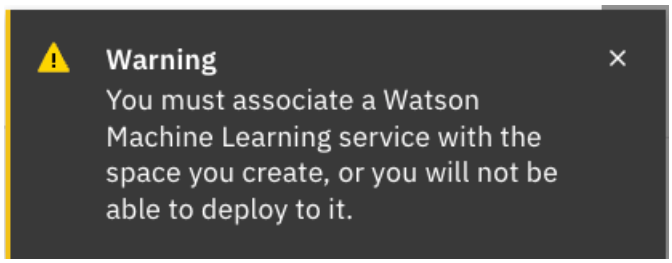
25. Click **Deployment space**. You must have a **Deployment space** to deploy your tuned model. If you do not have one already, you can create a new one. If you have one already available, you can select it and skip to Step 34, or you can create a new one.



26. There is no existing deployment space so click **Create a new deployment space**.



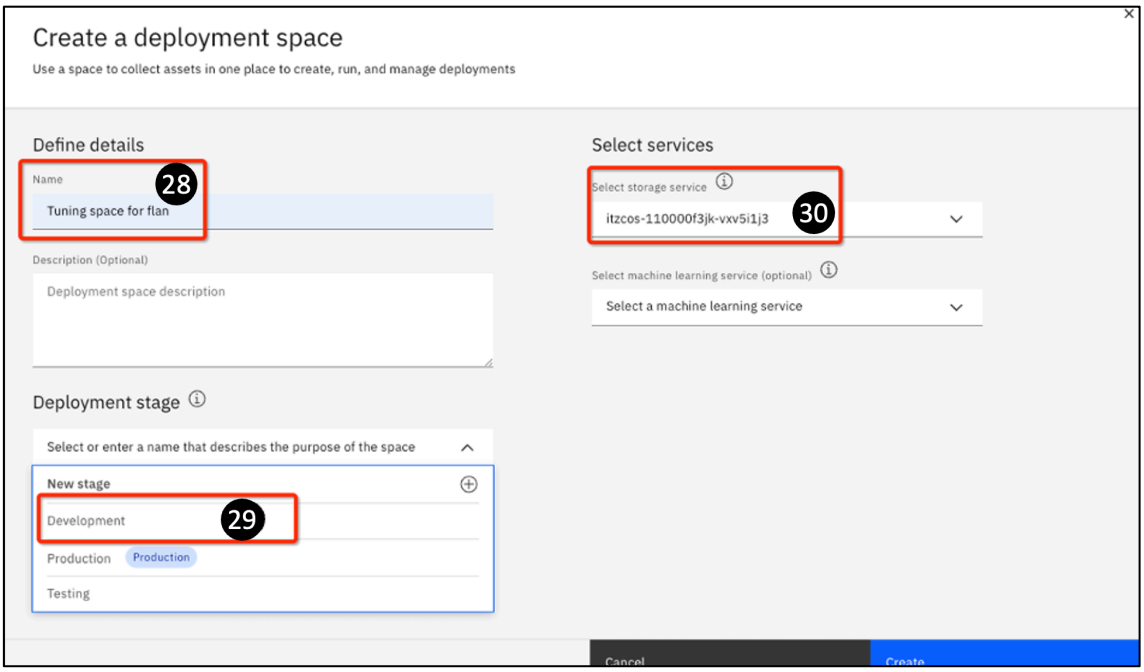
27. You will get the following warning. You can click anywhere, and it will disappear (it may take a couple of seconds – to ensure you have read it).



28. The **Create a deployment space** page opens. Provide a name for **Deployment space name**. Use any name you desire or enter **Tuning space for flan**.

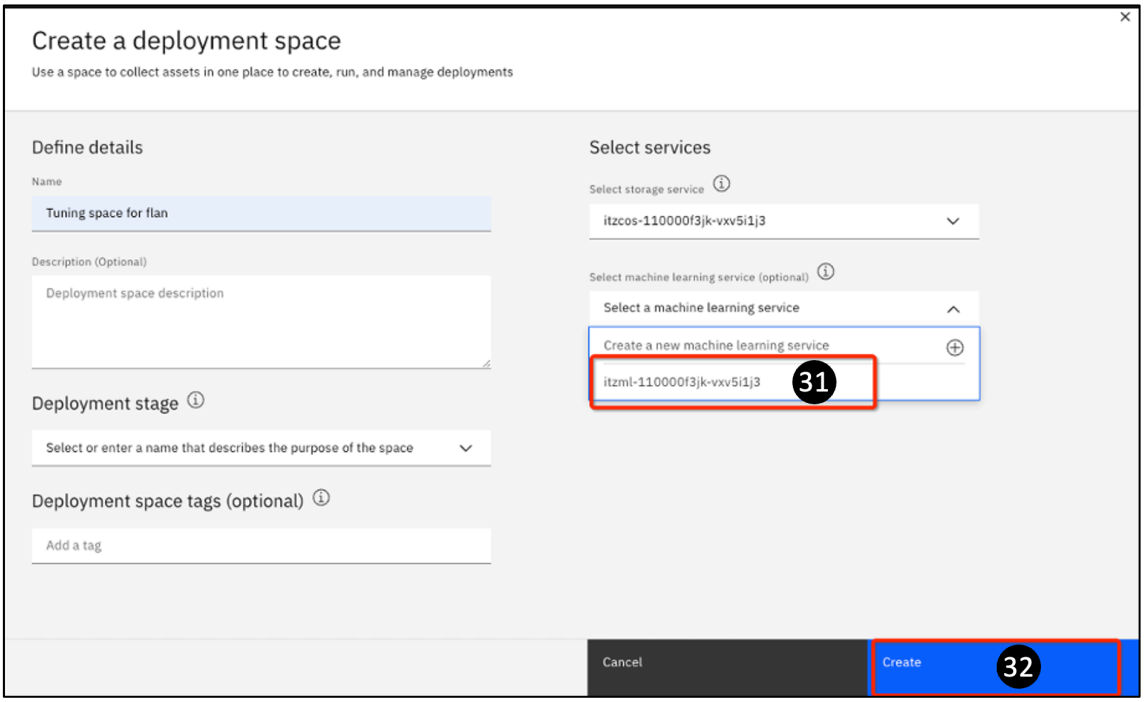
29. Click on the dropdown for **Deployment Stage** and select **Development**.

30. You need a storage device. If you are using a **TechZone** reservation one should be set up for you and is automatically selected. You can also create a different one and use that instead (if you have the necessary authority and access).

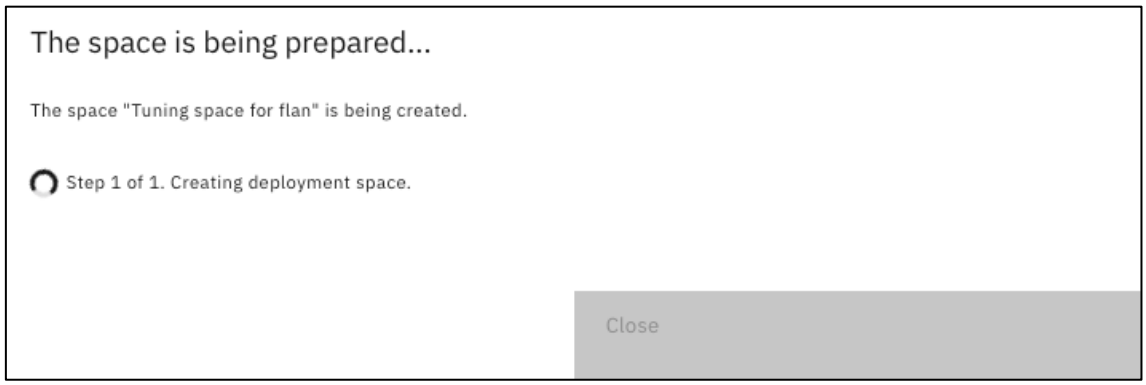


31. Click the **Select a machine learning service** pulldown. With a **TechZone** reservation, there is a machine learning service available, click to select it.

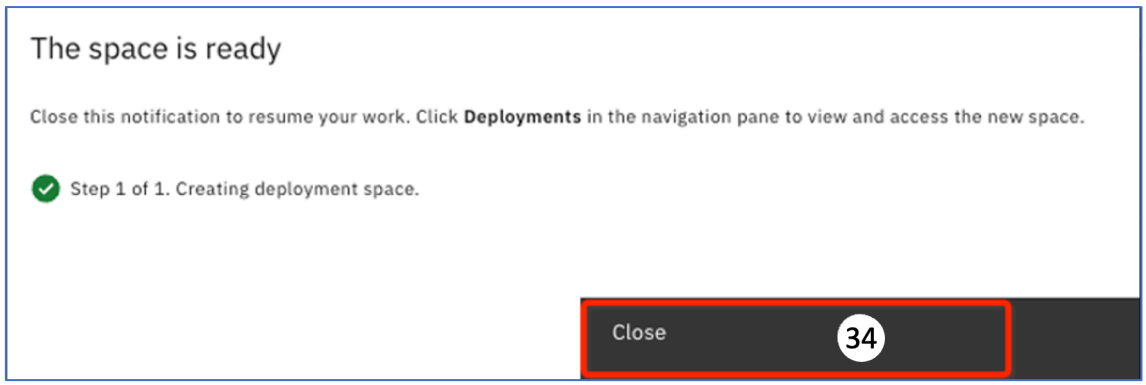
32. Click **Create**.



33. **The space is being prepared...** page opens

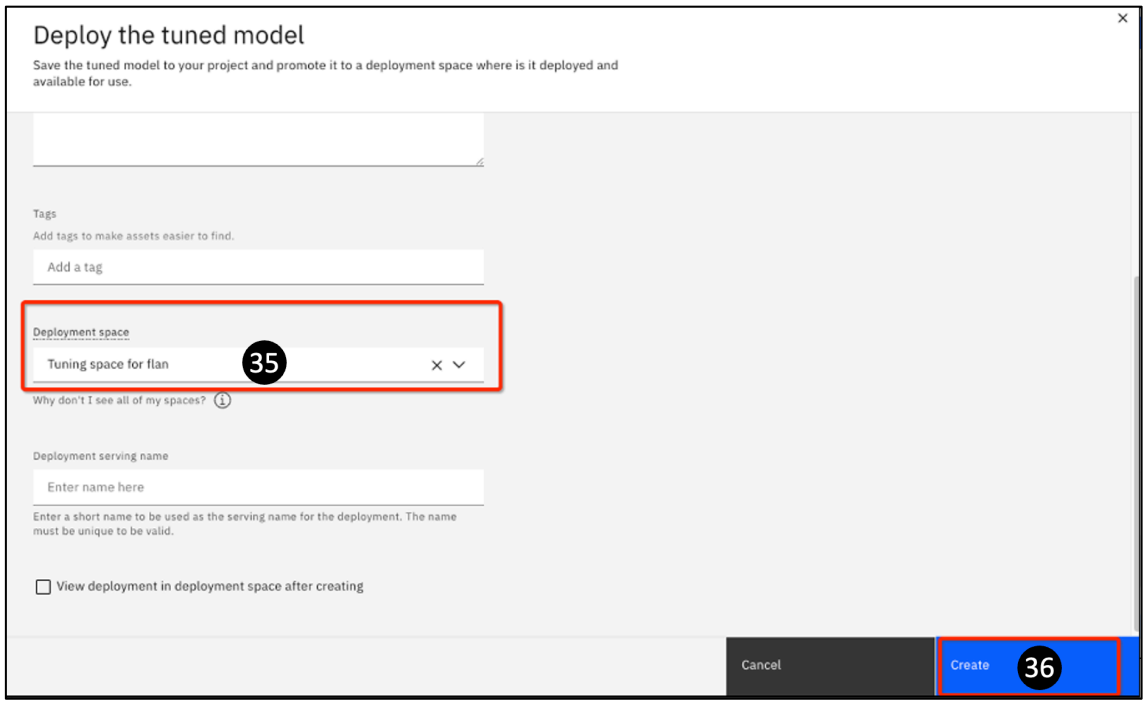


34. When it is completed, the title of the page changes to **The space is ready**. Click **Close.**

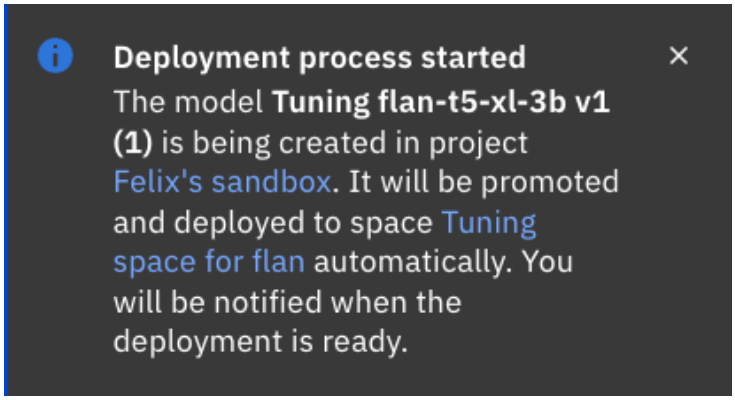


35. You are back to the **Deploy the tuned model** page. In this case, you have created your first **Deployment space** and the value **Tuning space for flan** is automatically filled in.

36. Click **Create**.



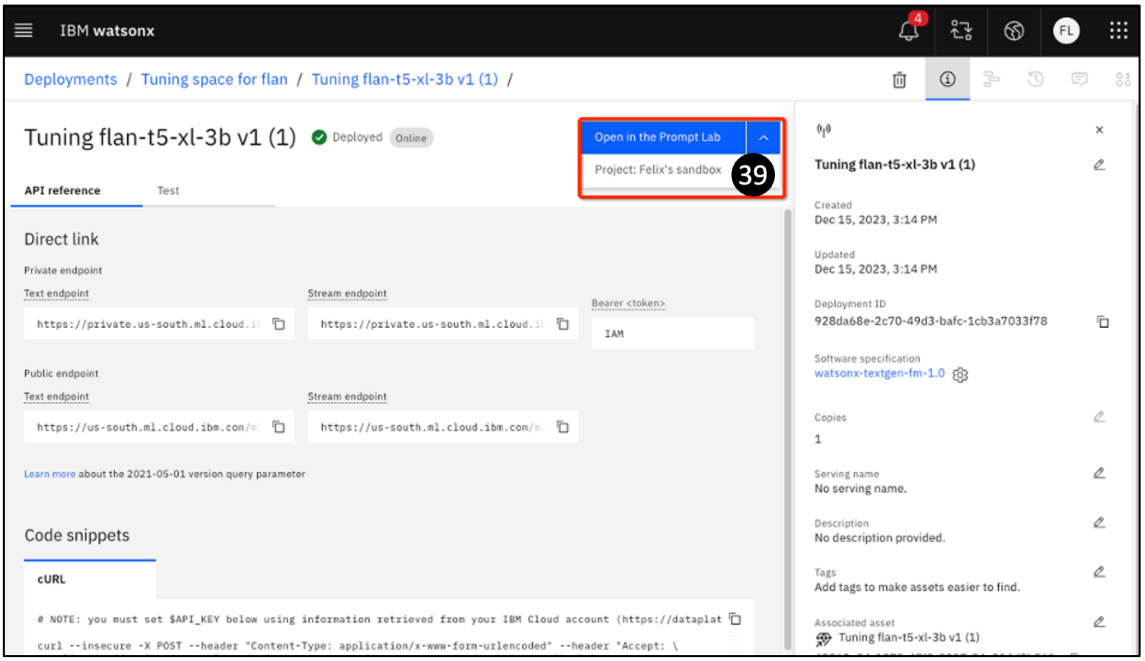
You get this message:



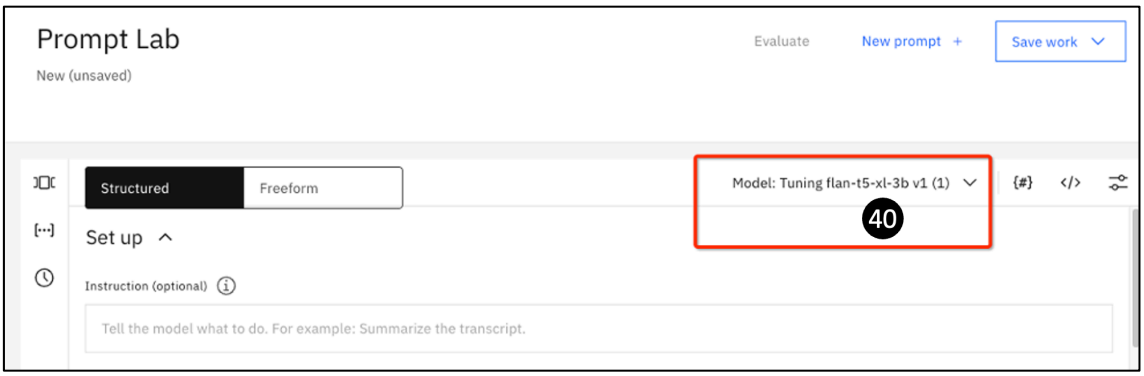
37. When completed, you will see that your tuned model is deployed. Click on this **Tuning flan-t5-xl-3b v1 (1)** model.

38. The **Tuning flan-t5-xl-3b v1 (1)** page opens with information you need to call this model (such as **Public** and **Private endpoints**).

39. Click on the **Open in the Prompt Lab** pulldown and select the project you want to use. In the example below, it is **Project: Felix's sandbox**.



40. The watsonx.ai **Prompt Lab** page opens and the **Tuning flan-t5-xl-3b v1 (1)** model is automatically selected.



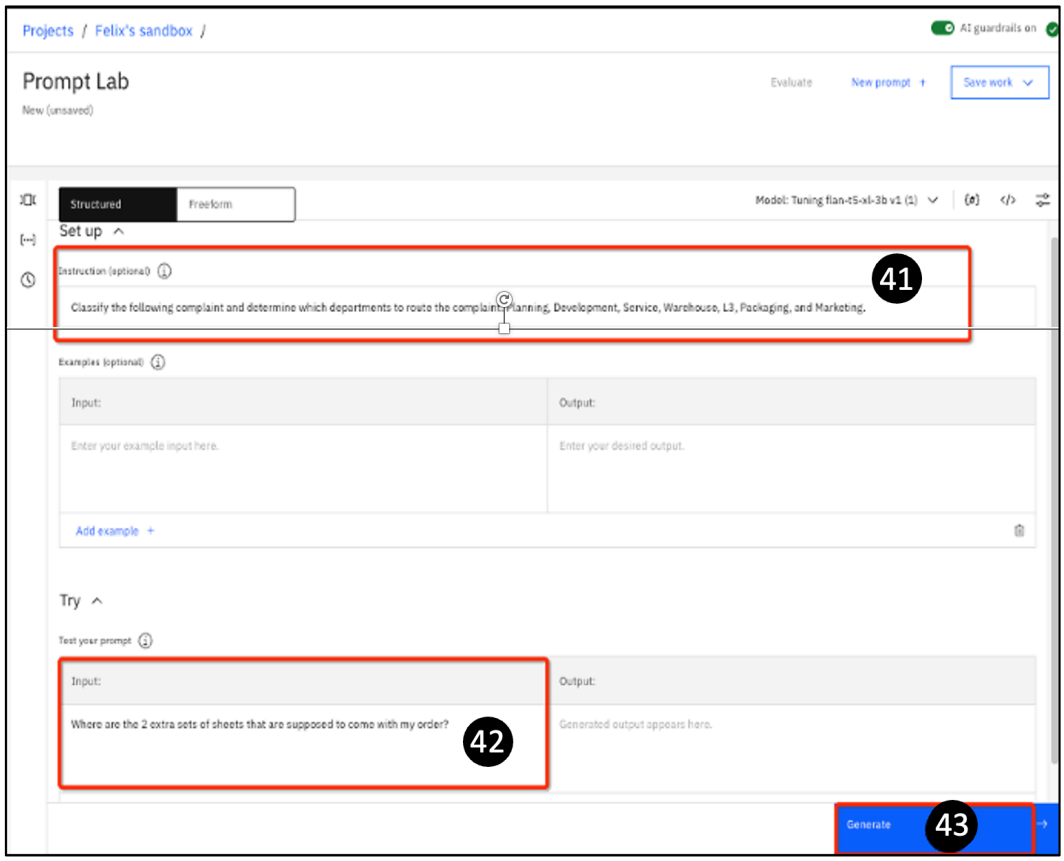
41. Paste the following text in the **Instruction** field:

    Classify the following complaint and determine which departments to route the complaint: Planning, Development, Service, Warehouse, L3, Packaging, and Marketing.

42. Enter the following text into the **Input** field in the **Try** section:

    Where are the 2 extra sets of sheets that are supposed to come with my order?

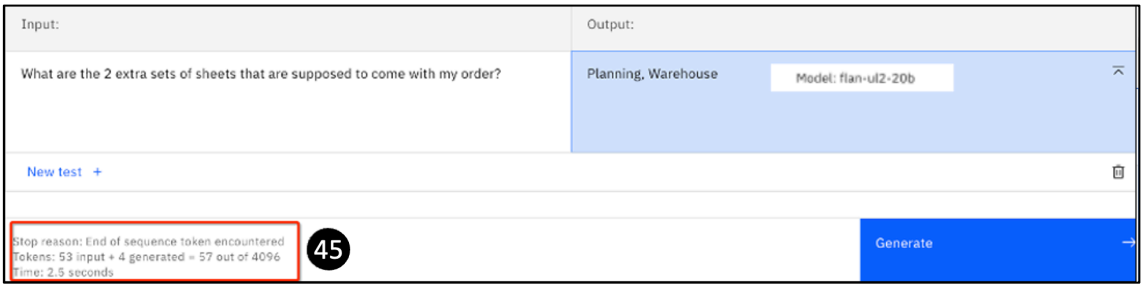
43. Click **Generate**.



44. This time, we see this completion:

And that completion of **Warehouse, Packaging** is exactly what you want to see.

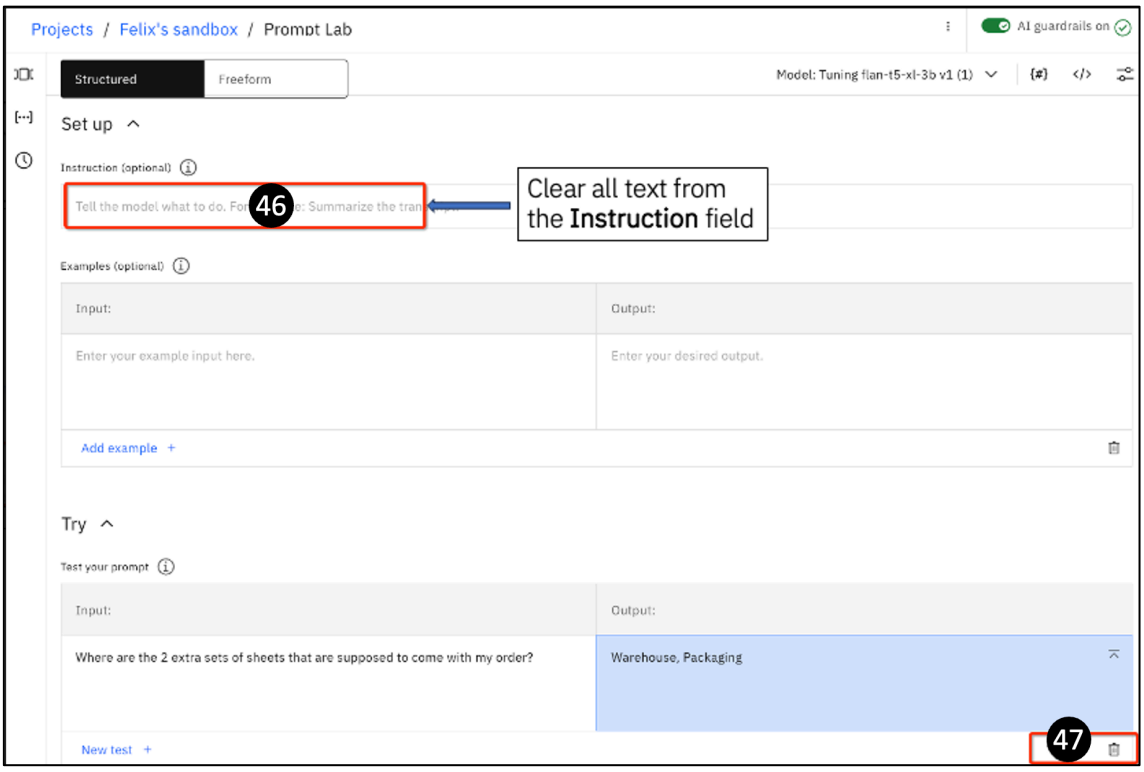
45. Note the information in the lower left corner:



The token count is 53 input + 4 generated. You will compare this count against earlier counts in the **Section summary**.

46. Remove the text in the **Instruction** field.

47. Clear the previous **Input** and **Output** by clicking on the **Garbage Can** (🗑) icon.

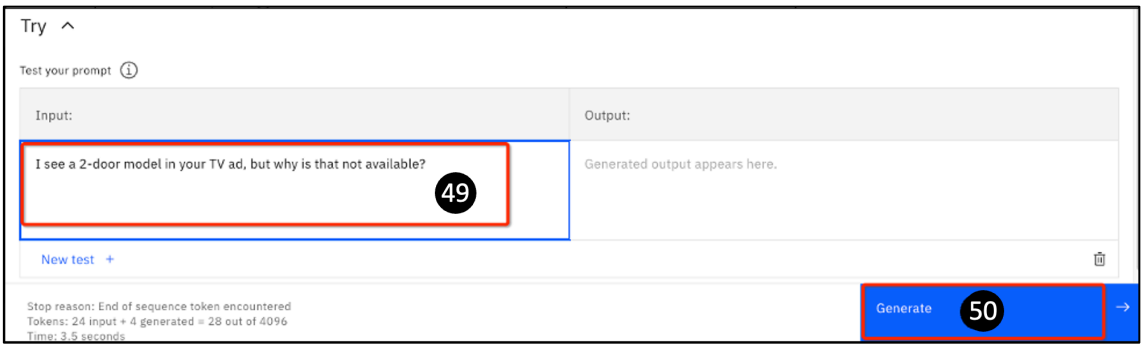


48. Click **New test +** to add a new set of **Input**/**Output** cells in the **Try** Section.

49. Try a different complaint. Enter the following into the **Input** field in the **Try** section:

```
I see a 2-door model in your TV ad, but why is that not available?
```

50. Click **Generate**.



51. The **Tuning flan-t5-xl-3b v1 (1)** model returns with a completion of **Planning, Marketing**. This is what you expected according to the business rules. The fact that it appears in the TV advertisement but is not available can be a marketing mistake. On the other hand, if this is truly a missing feature then clearly customers are looking for it, so Planning should be notified.

    Notes:

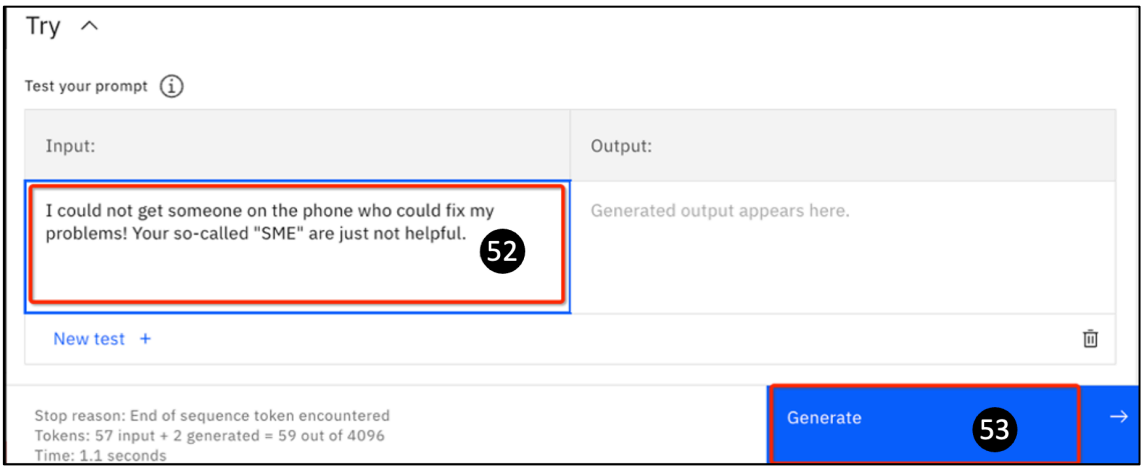
    - The base LLM **flan-t5-xl-3b** (you can try it) would just return **Marketing**, which is half the answer. There is just no way the base LLM can know the business logic and that multiple departments can be a perfectly valid (and desirable) answer.

    - You do not need to provide any text for **Instruction** as that information was already included when you did the prompt tuning in Section 6.2 Step 11. This is another advantage of prompt tuning over prompt engineering. With no need for **Instruction**, fewer tokens are consumed *every time* this tuned model is used. This makes it easier to use, and the cost savings will add up.

52. You can try one more. Repeat Steps 45-46 to clear the fields and add a new pair of **Input**/**Output** fields. Paste the following into the **Input** field.

    > I could not get someone on the phone who could fix my problems! Your so-called "SMEs" are just not helpful.

53. Click **Generate.**



54. The **Tuning flan-t5-xl-3b v1 (1)** model returns with the completion of **Service, L3**. This is again what is expected according to the business rules. This is clearly a Service issue, but with the comment on SME, the Level 3 support team needs to be notified as well.

    If you tried the base **flan-t5-xl-3b** model, it would respond with just **Service**. The issue is clearly one of service and the LLM can figure that out. What it does not have is the training from the labeled data that understand issues associated with "SME" are also L3 issues – per the company's process.

    Once again, there was no need to provide any instructions.

Section summary

- In this section, you tuned the **flan-t5-xl-3b** model with a set of labeled data which includes examples of complaints and where the business wants them to be routed.
- With a good set of data, you can train a model to address specific business requirements. It can return completions based on unique business logic that cannot be learned outside of a set of validated labeled data used to train the model.
- You can train the same model with a different set of labeled data to address different business problems.
- Compare tokens usage.
    - First, you can Re-run the first query "Where are the 2 extra sets of sheets that are supposed to come with my order?" – this time with no Instruction. Note the tokens usage.
    - The following table compares previously noted token usage information for the same query: "Where are the 2 extra sets of sheets that are supposed to come with my

order?”

Prompt engineering Section 7.2, Step 18	47 input + 2 generated = 49 out of 4096
3-shot prompting Section 7.3, Step 19	106 input + 2 generated = 108 out of 4096
Prompt tuned model Section 9.2, Step 42	53 input + 4 generated = 57 out of 4096
Prompt tuned model (with no **Instruction** text)	22 input + 4 generated = 26 out of 4096

This shows another advantage of **Prompt tuning**. It consumes not that much more input tokens than **Prompt engineering** and gets the proper results. It is almost half as costly (in terms of tokens consumed) as a **3-shot prompting** that did not work. The difference is especially pronounced in the case where you remove the **Instruction** text.

A one-time tuning can outperform at a lower cost than multi-shot prompting. In addition, multi-shot prompting only works for a particular prompt, and it may not work for a different prompt. Whereas **Prompt tuning** (as shown in the lab) can work for a whole class of problems.

6.3 The importance of data in tuning

Download a different set of labeled data Call center complaints v2 (again, ensure that the file extension is in lowercase **.jsonl**). You will repeat all the Steps 1-37 from Section 6.2 with several minor changes.

- **Step 2**
  Use a different **Name**, for example: **Tuning flan-t5-xl-3b v2**.
- **Step 11**
  Use the **Call center complaints v2.jsonl** file
- **Step 23**
  As you have already created a **Deployment space** in Section 6.2, you can re-use it. There is no need to create a new **Deployment space**.
- **Steps 34, 35, 36, 37**
  With this new tuning, instead of **Tuning flan-t5-xl-3b v1 (1)** model you will have **Tuning flan-t5-xl-3b v2 (1)** instead.

Now that the model is tuned and available, you will retry some of the LLM completions.

1. Ensure that you are using the **flan-t5-xl-3b v2 (1)** model, NOT the **flan-t5-xl-3b v1 (1)** model.

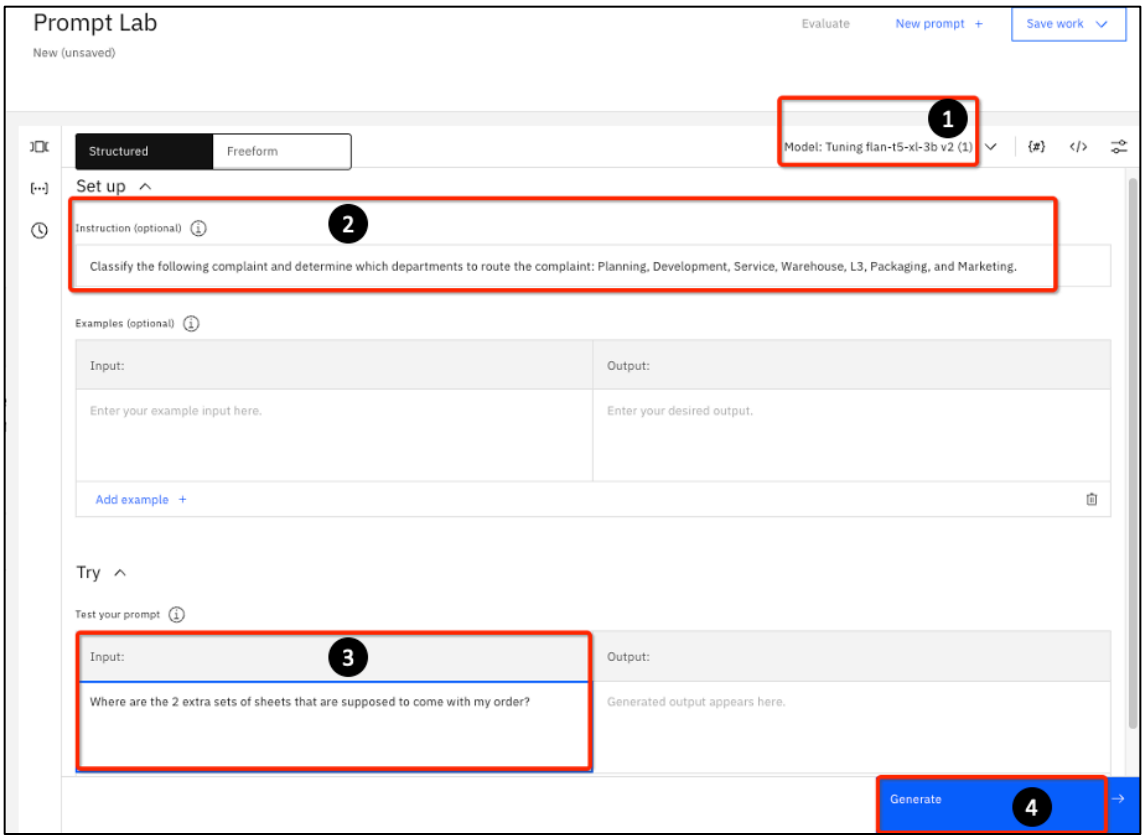
2. Enter the following into the **Instruction** field:

   ```
   Classify the following complaint and determine which departments to route the complaint: Planning, Development, Service, Warehouse, L3, Packaging, and Marketing.
   ```

3. Enter the following into the **Input** field in the **Try** section:

   ```
   Where are the 2 extra sets of sheets that are supposed to come with my order?
   ```

4. Click **Generate**.



With this new tuning – the completion now is **Warehouse** (instead of **Warehouse, Packaging**).

5. Next, use the following **Input**

> I see a 2-door model in your TV ad, but why is that not available?

Click **Generate** and the completion is **Planning, Warehouse** (instead of **Planning, Marketing**).

6. Next, use the following **Input**

> I could not get someone on the phone who could fix my problems! Your so-called "SME" are just not helpful.

Click **Generate** and the completion is **Service, L3**.

The set of labeled data in **Call center complaints v2.jsonl** is a set of valid complaints and classifications by the business. However, it is not as good a set of training data as the initial set. Why is that?

6.3.1 An anatomy of a set of labeled data

It is important to understand what you are trying to accomplish with prompt tuning. In this case, you can make some simple assumptions:

- The LLM has a reasonable understanding of **Planning**, **Development**, **Service**, **Warehouse**, **Packaging**, and **Marketing**. The only item that may be a bit unclear could be **L3**. A human (and a well-rounded LLM) could read in context and deduce this to be Level 3 Subject Matter Expert (SME) support. You may need to include this information in some training.

- The LLM would be reasonably capable of matching the complaints to a single category.

What you would need to teach the model, however, is the following:

- Complaints may match multiple categories.
  This means a good representation of each of the possible category outcomes in the list.

- Business rules on how to match a complaint to specific categories
  This means a good representation of complaints that map to non-trivial/multiple categories as determined by business rules.

- Complaints that should involve L3, including samples of complaints that:
  - Map to Service and L3
  - Map to just L3

A labeled data set that satisfies the above will provide better training for the model.

In this case, both sets of training data used in Section 6.2 (**Call center complaints.jsonl**) and Section 6.3 (**Call center complaints v2.jsonl**) contain valid business data but generate different completions.

It is useful to look into some details of the data provided and see what the differences are and how that might have influenced the decision-making process of the subsequent models in providing completions.

First query

- **Input**: Where are the 2 extra sets of sheets that are supposed to come with my order?
- **Tuning flan-t5-xl-3b v1** prompt tuned with **Call center complaints v1.jsonl**.

  Model completion: **Warehouse, Packaging** – this is the expected completion.
- **Tuning flan-t5-xl-3b v2** prompt tuned with **Call center complaints v2.jsonl**.

  Model completion: **Warehouse**.

The following table shows details of the sample data used to train the 2 models.

Total number of entries for both input files is 200	Call center complaints.jsonl	Call center complaints v2.jsonl
Number of entries with just **Warehouse** as output	3	8
Number of entries with **Warehouse, Packaging** as output	16	5
Number of entries with **Warehouse** and something else (not **Packaging)** as output	22	10

These numbers tell an interesting story.

- **Tuning flan-t5-xl-3b v1** was tuned with **Call center complaints.jsonl.** This jsonl file contains only 3 entries with just **Warehouse** as the routing target of the complaint. There are 16 examples of complaints going to **Warehouse, Packaging**, and 22 more going to **Warehouse** and some other categories. This teaches the model that:
    - **Warehouse**, **Packaging** is a common occurrence, and there are 16 examples (enough to show some patterns).
    - **Warehouse** is often paired with some other categories (aside from **Packaging)**. In contrast, it is less likely to stand on its own.

  With this knowledge, the tuned LLM is more likely to consider additional categories in the completion once it identifies a complaint with **Warehouse.** It also has enough education to know when to use **Warehouse, Packaging**. It is then able to correctly provide the proper completion (**Warehouse, Packaging)** for the first query: “Where are the 2 extra sets of sheets that are supposed to come with my order?”
- **Tuning flan-t5-xl-3b v2** was tuned with **Call center complaints v2.jsonl**. The v2 file in contrast does not have as good of a distribution. There are 8 examples with just **Warehouse** as completion, more so than the 5 examples where **Warehouse, Packaging** is the expected completion. This is saying **Warehouse** is more likely to be the completion, which happened when you used **Tuning flan-t5-xl-3b v2**.

Second query

- **Input**: I see a 2-door model in your TV ad, but why is that not available?
- **Tuning flan-t5-xl-3b v1** completion: **Planning, Marketing** – this is the expected completion.
- **Tuning flan-t5-xl-3b v2** completion: **Planning**, **Warehouse**

This difference here is even more pronounced with similar data points.

Total number of entries for both input files is 200	Call center complaints.jsonl	Call center complaints v2.jsonl
Number of entries with just **Planning** as output	17	16
Number of entries with **Planning, Marketing** as output	18	3
Number of entries with **Planning** and something else (not **Marketing**) as output	64	18

- **Tuning flan-t5-xl-3b v1** was tuned with **Call center complaints.jsonl.** This jsonl file has 17 examples where **Planning** is the completion. However, there are 18 examples where **Planning** is paired with **Marketing**, with another 64 entries where **Planning** is paired with other categories as the expected completion.
    - This says that **Planning, Marketing** is just as likely to appear as **Planning** on its own.
    - **Planning** is much more likely to be paired with something else than appearing on its own.

  Using **Call center complaints.jsonl** for prompt tuning, the resulting **Tuning flan-t5-xl-3b v1** model finds a good balance and properly returns a completion of **Planning, Marketing**.
- **Tuning flan-t5-xl-3b v2** was tuned with **Call center complaints v2.jsonl**. The v2 file in contrast does not have as good of a distribution. There are 16 entries where **Planning** is the sole expected completion. There are only 3 examples of **Planning, Marketing,** and another 18 examples where **Planning** is paired with something else as an output. The message/knowledge being conveyed here is that:
    - **Planning** – the likelihood of this appearing on its own is close to it being paired with something else in a completion.
    - When it does pair up with something in completion, the probability that it would be **Marketing** is low. There are 10 examples of **Planning, Warehouse**. This is why this tuned model returned that as a more probable result.

  So, unless the **Input** is very close to the 3 **Planning, Marketing** examples, the model would likely not consider that as a good completion.

Section summary

- Prompt Tuning is a powerful tool to help train a model to perform specific tasks (especially when there are business rules that are not known outside of the company).
- It is important to pay attention to the content of the training data. LLMs are quite smart, and the labeled data input does not determine how the LLM will perform a completion, but it does help to steer it in a certain direction. As such, the data used mustn't inadvertently introduce new biases to the model.
- Take care in creating the sample data set.
    - Spend time with the client to come up with a set of data that can augment the knowledge base of the model. This depends on the downstream tasks you need the LLM to perform.
    - The key is to highlight what the LLM does not naturally do (such as identifying multiple categories in a classification task), or rules (such as business rules) that it could not have learned with its training.

      For example, in Section 6.2, you want to steer the LLM toward understanding that multiple categories in the completion are proper; you want the LLM to know certain types of complaints should be directed to specific groups based on business logic. Your sample set should include plenty of examples of multiple-category completions that illustrate how complaints with specific wordings are routed. The input **Call center complaints.jsonl** is a fairly good example.
    - In contrast, (similar to what you saw in Section 6.3) a straight percentage representation of the actual data may not be useful. If you have 1 million complaints in your database and 900,000 of those have a single completion, then a % representation means you will create a 200-sample dataset with 180 entries with a single completion output, and only 20 entries with multiple entries output. This may be valid but will have similar effects as **Call center complaints v2.jsonl**. It will steer the LLM towards a high probability of providing a single output completion. This would have caused defects in the purpose of the prompt tuning.

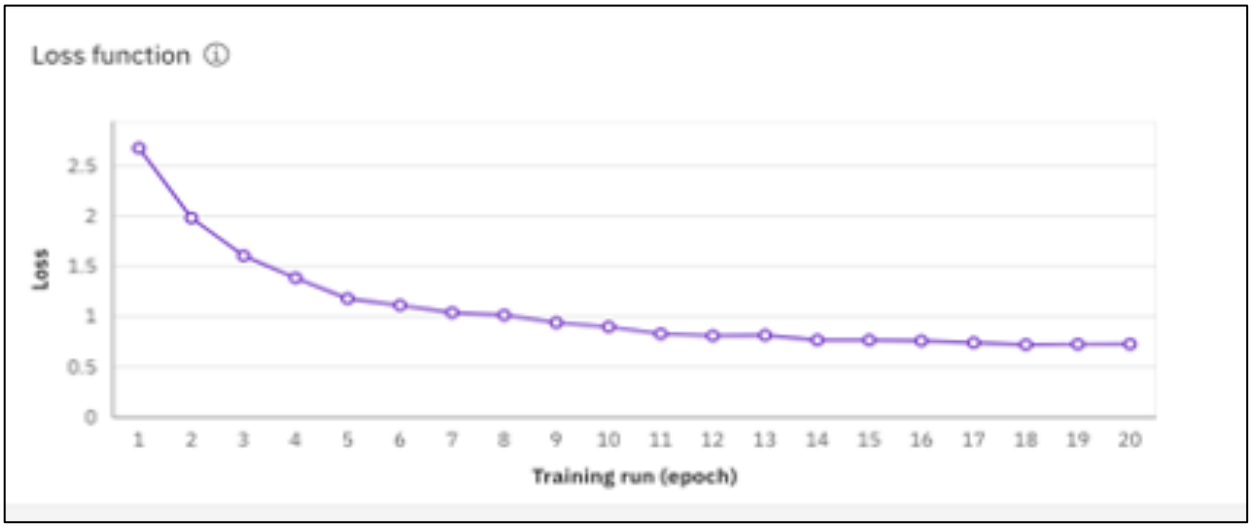
Appendix A. Learning Rate

This is not meant to be a thorough discourse of Learning Rate, but sufficient information to understand why you may need to tune the Learning Rate parameter when you are performing prompt tuning.

As seen in Step 22 of Section 6.2, part of prompt tuning is the calculation of the **Loss function.** At a simple level, the **Loss function** measures how well the LLM models your dataset. The closer the LLM's completion is to the actual desired output, the smaller the **Loss**. So, in prompt tuning, you want to find the lowest point of the **Loss function**. Ideally, one would like to see it reach zero – but that is difficult (impossible almost).

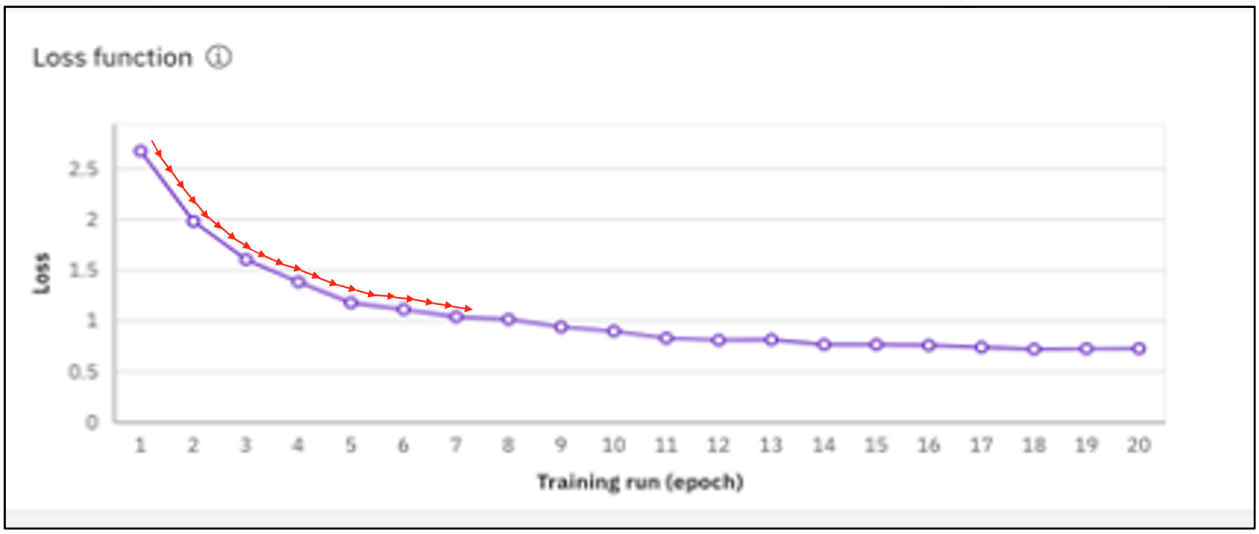
This is where **Number of epochs** and **Learning rate** come in. The **Number of epochs** is how many times to cycle through the training data. While in general the more epochs the better, there will be a point of diminishing return where the cost of another epoch brings no further improvement.

Here is the graph of the **Loss function** from Step 22 of Section:



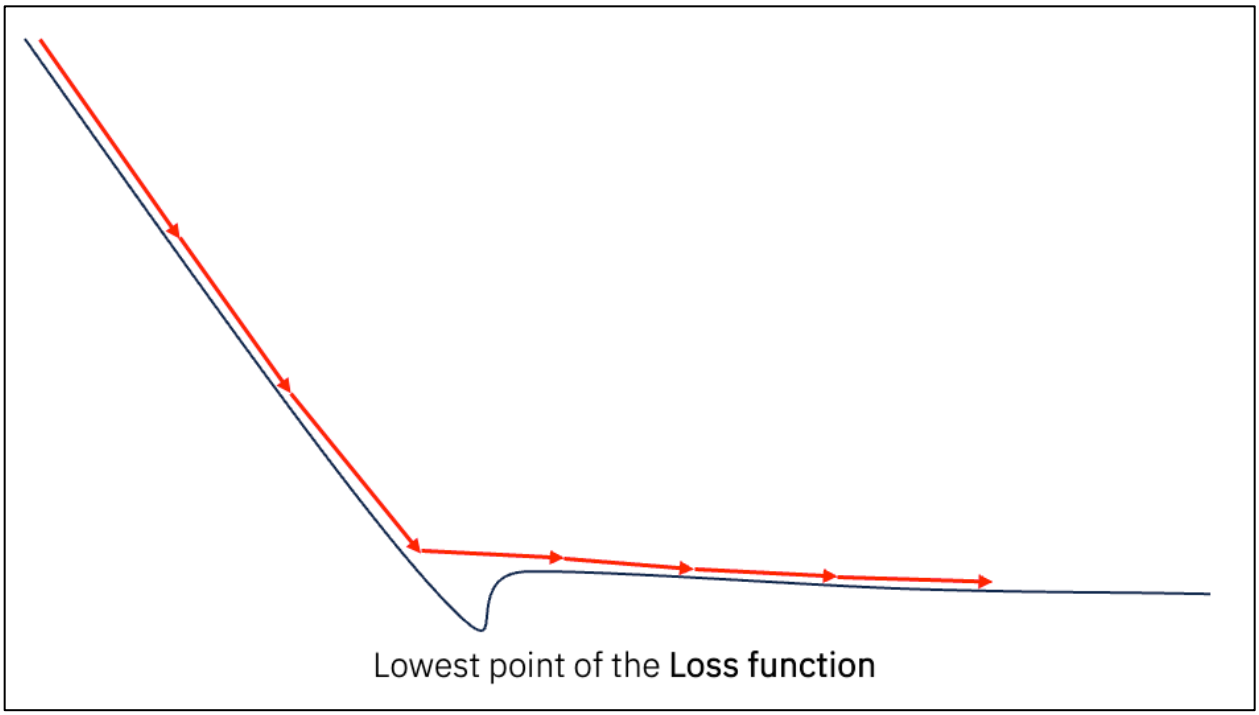
In this function, the lowest value seems to be around 0.75. The distance between each dot can be thought of as the **Learning rate**.

If the **Learning rate** value is too low, then even after 20 epochs the **Loss function** might have only reached a value of 1 (see the 20 red arrows below). This means that the LLM is not close to providing the best completion for the test data.



One might be tempted to use a large **Learning rate**. This may not be a bad idea if the **Loss function** is shaped as above – and ever dipping line. A larger **Learning rate** may get to the lowest point sooner (or as low as the **Number of epochs** allows).

However, a high **Learning rate** means you are allowing the model algorithm to accept large changes. You can "overshoot" the lowest point of the **Loss function**. For example. If the actual **Loss function** looks like the black curve below.



The red arrows reflect a large **Learning rate,** so it progresses quickly. However, because of the large steps it completely missed the lowest point of the **Loss function**. This means that the algorithm change to the LLM will not be optimal, even though it may be "not bad".

The natural question arises of how to set the **Learning rate**. Here is a suggestion:

- Start with the default value.
- If the **Loss function** is levelling off early (as in the example in Section 6.2), try a smaller **Learning rate** to see if there might be hidden valleys.
- If the **Loss function** is still showing continued decline after 20 epochs, try a larger **Learning rate** or a higher **Number of epochs**.

Always keep in mind – the **Loss function** is calculated based on the data you put in. So, while you might get to a levelling off at a low value, it just means the LLM is now able to perform generations reflective of the labeled data set. It does NOT necessarily mean it is optimally tuned to your entire data set. Simply think of the **Loss function** for Section 6.3 using **Call center complaints v2.jsonl**. It too would have been leveling off. However, it certainly does not perform as well as the model trained on **Call center complaints.jsonl**.

Appendix B. Revision History

Date	Changes
-	Original version.