Fine-tune a language model with Microsoft Foundry

1. Introduction

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/1-introduction

Introduction

Completed

- 1 minute

Language models are pretrained models that provide you with a great starting point. By using one of the available base or foundation models, you can save time and effort as you need less data to train a model for your specific use case.

Imagine you're a developer working for a travel agency. When customers use your chat application to get help with their travel-related questions, you want the responses to be in a specific format and style. Your company has a specific tone of voice that resonates with your client base. The marketing department finds it important that the chat application is aligned with your company's tone of voice too.

There are various strategies to optimize the model's behavior and the performance of your chat application. One strategy is to fine-tune a language model, which you can then integrate with your chat application. The benefit of fine-tuning over training your own language model, is that you need less time, compute resources, and data to customize the model to your needs.

In this module, you learn how to fine-tune a base model from the model catalog in the Microsoft Foundry portal, that you can then integrate in a chat application.

2. Understand when to fine-tune a language model

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/2-understand-finetune

Understand when to fine-tune a language model

Completed

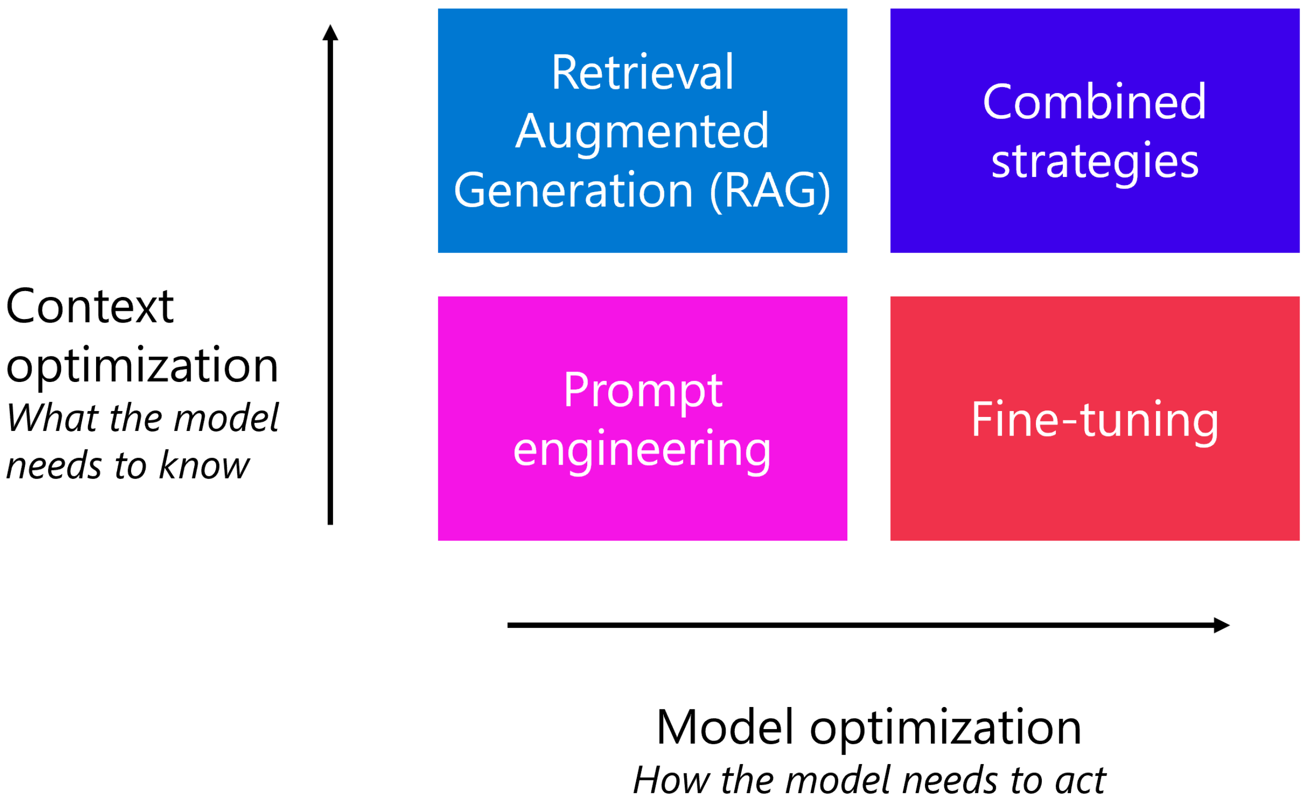
- 5 minutes

Before you start fine-tuning a model, you need to have a clear understanding of what fine-tuning is and when you should use it.

When you want to develop a chat application with Microsoft Foundry, you can use prompt flow to create a chat application that is integrated with a language model to generate responses. To improve the quality of the responses the model generates, you can try various strategies. The easiest strategy is to apply **prompt engineering**. You can change the way you format your question, but you can also update the **system message** that is sent along with the prompt to the language model.

Prompt engineering is a quick and easy way to improve *how the model acts*, and *what the model needs to know*. When you want to improve the quality of the model even further, there are two common techniques that are used:

- **Retrieval Augmented Generation** (**RAG**): Ground your data by first retrieving context from a data source before generating a response.
- **Fine-tuning**: Train a base language model on a dataset before integrating it in your application.


Retrieval Augmented Generation (RAG)
Combined strategies
Prompt engineering
Fine-tuning
Context optimization
What the model needs to know
Model optimization
How the model needs to act

RAG is most commonly applied when you need the model's responses to be factual and *grounded* in specific data. For example, you want customers to ask questions about hotels that you're offering in your travel booking catalog. On the other hand, when you want the model to behave a certain way, fine-tuning can help you achieve your goal. You can also use a combination of optimization strategies, like RAG *and* a fine-tuned model, to improve your language application.

How the model needs to act mostly relates to the style, format, and tone of the responses generated by a model. When you want your model to adhere to a specific style and format when responding, you can instruct the model to do so through prompt engineering too. Sometimes however, prompt engineering might not lead to consistent results. It can still happen that a model *ignores* your instructions and behaves differently.

Within prompt engineering, a technique used to "*force*" the model to generate output in a specific format, is to provide the model with various examples of what the desired output might look like, also known as **one-shot** (one example) or **few-shot** (few examples). Still, it can happen that your model doesn't always generate the output in the style and format you specified.

To maximize the **consistency of the model's behavior**, you can **fine-tune a base model** with your own training data.

3. Prepare your data to fine-tune a chat completion model

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/3-prepare-data

Prepare your data to fine-tune a chat completion model

Completed

- 5 minutes

Fine-tuning involves combining a suitable *foundation* model to use as a base, and with a set of *training data* that includes example prompts and responses that the model can learn from.

When you decide you want to fine-tune a language model, you need to identify the dataset you can use to fine-tune your language model.

Similar to any machine learning model, the quality of the dataset has a large effect on the quality of your model. Though you need less data than when you would train a language model from scratch, you still might need enough data to maximize the consistency of your desired model's behavior. How much data you need depends on your use case.

When you fine-tune a language model for chat completion, the data you use to fine-tune a model is a collection of sample conversations. More specifically, the data should contain three components:

- The system message
- The user message
- The assistant's response

The three variables come together in a JSON Lines or JSONL file. For example, one line in such a dataset might look like:

```
{"messages": [{"role": "system", "content": "You are an Xbox customer support agent whose primary
```

The dataset should show the model's ideal behavior. You can create this dataset based on the chat history of a chat application you have. A few things to keep in mind when you use real data is to:

- Remove any personal or sensitive information.
- Not only focus on creating a large training dataset, but also ensure your dataset includes a diverse set of examples.

You can include multiple turns of a conversation on a single line in the dataset. If you want to fine-tune only on specific assistant messages, you can optionally use the `weight` key-value pair. When the weight is set to 0, the message is ignored, when you set to 1, the message is included for training.

An example of a multi-turn chat file format with weights:

```
{"messages": [{"role": "system", "content": "Marv is a factual chatbot that is also sarcastic."},
```

When preparing your dataset to fine-tune a language model, you should understand your desired model behaviors, create a dataset in JSONL format, and ensure the examples you include are high quality and diverse. By preparing your dataset, you have a higher chance that the fine-tuned model improves your chat application's performance.

4. Explore fine-tuning language models in Microsoft Foundry portal

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/4-finetune-model

Explore fine-tuning language models in Microsoft Foundry portal

Completed

- 8 minutes

When you want to fine-tune a language model, you can use a base or foundation model that is already pretrained on large amounts of data. There are many foundation models available through the model catalog in Microsoft Foundry. You can fine-tune base models on various tasks, like text classification, translation, or chat completion.

When you want to use a fine-tuned model to generate responses in a chat application, you need to use a base model that can be fine-tuned on a chat completion task. The Microsoft Foundry model catalog allows you to filter based on fine-tuning tasks to decide which base model to select. You can, for example, select a GPT-4 or Llama-2-7b model to fine-tune on your own training data.

To fine-tune a language model from Microsoft Foundry's model catalog, you can use the user interface provided in the portal.

Select the base model

When you navigate to the model catalog in the Microsoft Foundry portal, you can explore all available language models.

> **Note**
>
> Though all available language models will appear in the Microsoft Foundry model catalog, you might not be able to fine-tune the model you want depending on the available quota. Ensure the model you want to fine-tune is available in the region you created your AI hub in.

You can filter the available models based on the task you want to fine-tune a model for. Per task, you have several options for foundation models to choose from. When deciding between foundation models for a task, you can examine the description of the model, and the referenced model card.

Some considerations you can take into account when deciding on a foundation model before fine-tuning are:

- **Model capabilities**: Evaluate the capabilities of the foundation model and how well they align with your task. For example, a model like BERT is better at understanding short texts.
- **Pretraining data**: Consider the dataset used for pretraining the foundation model. For example, GPT-2 is trained on unfiltered content from the internet that can result in biases.
- **Limitations and biases**: Be aware of any limitations or biases that might be present in the foundation model.
- **Language support**: Explore which models offer the specific language support or multilingual capabilities that you need for your use case.

> **Tip**
>
> Though the Microsoft Foundry portal provides you with descriptions for each foundation model in the model catalog, you can also find more information about each model through the respective model card. The model cards are referenced in the overview of each model and hosted on the website of Hugging Face.

Configure the fine-tuning job

To configure a fine-tuning job using the Microsoft Foundry portal, you need to do the following steps:

1. Select a base model.
2. Select your training data.
3. *(Optional)* Select your validation data.
4. Configure the advanced options.

When you submit a model for fine-tuning, the model is further trained on your data. To configure the fine-tuning or training job, you can specify the following advanced options:

Name	Description
batch_size	The batch size to use for training. The batch size is the number of training examples used to train a single forward and backward pass. In general, larger batch sizes tend to work better for larger datasets. The default value and the maximum value for this property are specific to a base model. A larger batch size means that model parameters are updated less frequently, but with lower variance.
learning_rate_multiplier	The learning rate multiplier to use for training. The fine-tuning learning rate is the original learning rate used for pretraining multiplied by this value. Larger learning rates tend to perform better with larger batch sizes. We recommend experimenting with values in the range 0.02 to 0.2 to see what produces the best results. A smaller learning rate can be useful to avoid overfitting.
n_epochs	The number of epochs to train the model for. An epoch refers to one full cycle through the training dataset.
seed	The seed controls the reproducibility of the job. Passing in the same seed and job parameters should produce the same results, but can differ in rare cases. If a seed isn't specified, one is generated for you.

After you submit the fine-tuning job, a job will be created to train your model. You can review the status of the job while it's running. After the job is completed, you can review the input parameters when you want to understand how the fine-tuned model was created.

If you added a validation dataset, you can review the model's performance by exploring how it performed on your validation dataset.

Alternatively, you can always deploy a fine-tuned model. After deploying the model, you can test it to assess its performance. When you're satisfied with your fine-tuned model, you can integrate the deployed model with your chat application.

5. Exercise - Fine-tune a language model

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/5-exercise

Exercise - Fine-tune a language model

Completed

- 60 minutes

Now, it's your chance to explore how to fine-tune a foundation model from the model catalog using the Microsoft Foundry portal.

Note

To complete this lab, you will need an Azure subscription in which you have administrative access.

Launch the exercise and follow the instructions.

Launch Exercise

6. Module assessment

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/6-knowledge-check

Module assessment

Completed

- 3 minutes

7. Summary

https://learn.microsoft.com/en-us/training/modules/finetune-model-copilot-ai-studio/7-summary

Summary

Completed

- 1 minute

In this module, you learned:

- Understand when to fine-tune a model.
- Prepare your data to fine-tune a chat completion model.
- Fine-tune a base model in the Microsoft Foundry portal.

Learn more

- Customize a model with fine-tuning
- Fine-tune models using standard deployments in Microsoft Foundry
- Microsoft Foundry Discord
- Microsoft Foundry Developer Forum