Applied Deep Learning

5350 Retrieval Augmented Generation (RAG)

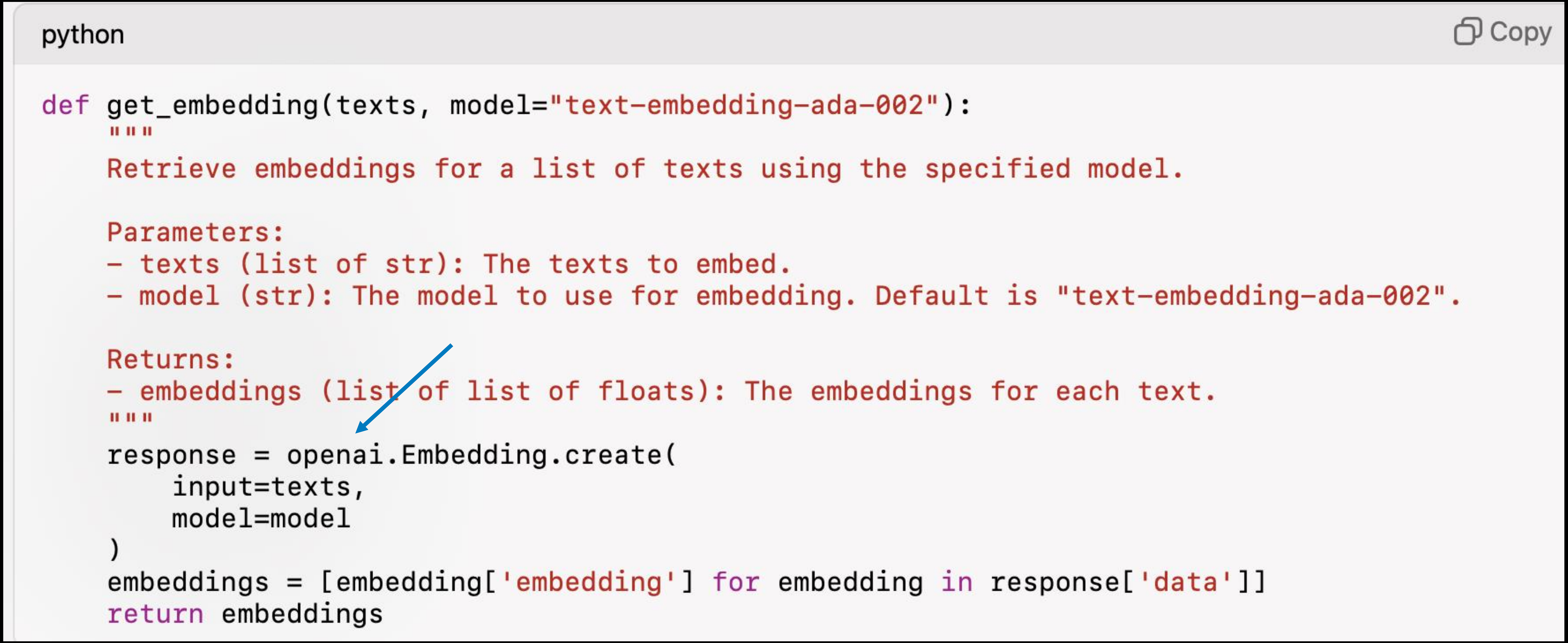
portilla@gmail.com

Jesus.portilla@Colorado.edu

https://github.com/iportilla/RAG

LLM Shortcomings

write a **get_embedding** function in python using the latest azure openai client


python

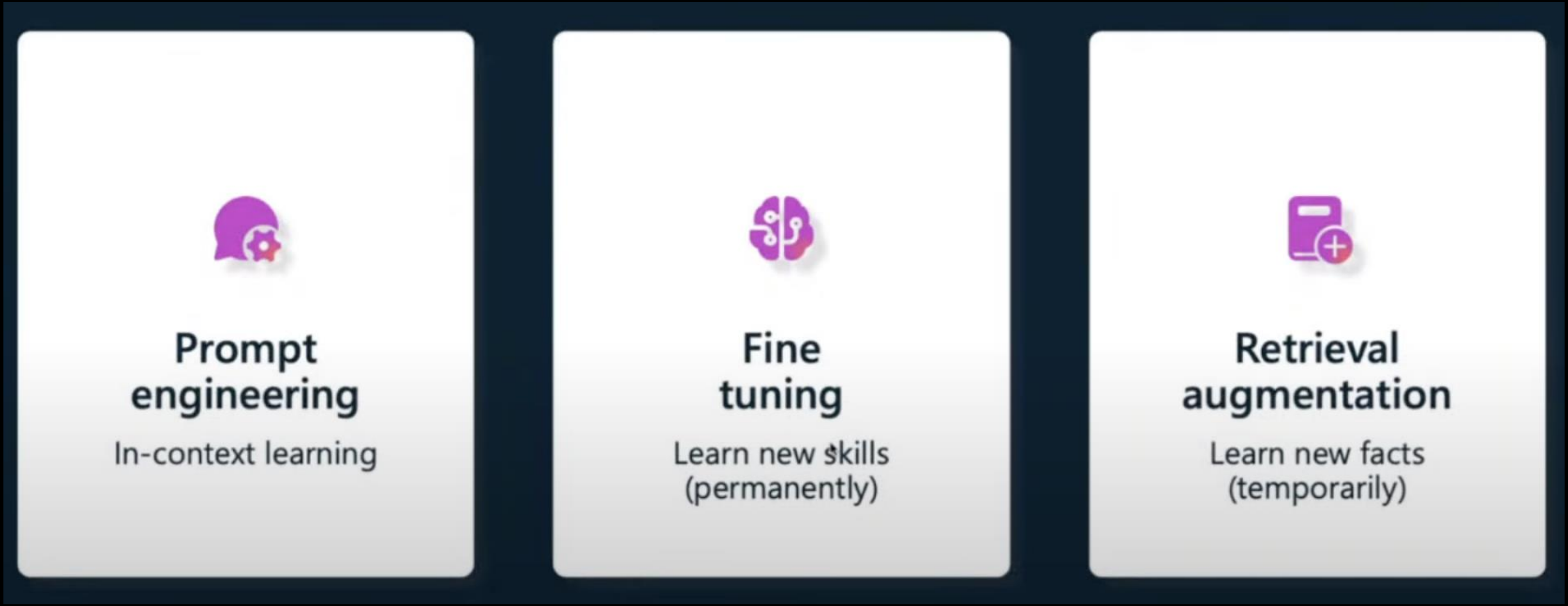
Copy

```python
def get_embedding(texts, model="text-embedding-ada-002"):
    """
    Retrieve embeddings for a list of texts using the specified model.

    Parameters:
    - texts (list of str): The texts to embed.
    - model (str): The model to use for embedding. Default is "text-embedding-ada-002".

    Returns:
    - embeddings (list of list of floats): The embeddings for each text.
    """
    response = openai.Embedding.create(
        input=texts,
        model=model
    )
    embeddings = [embedding['embedding'] for embedding in response['data']]
    return embeddings
```

Incorporate Domain Knowledge with LLM



Prompt engineering

In-context learning

Fine tuning

Learn new skills (permanently)

Retrieval augmentation

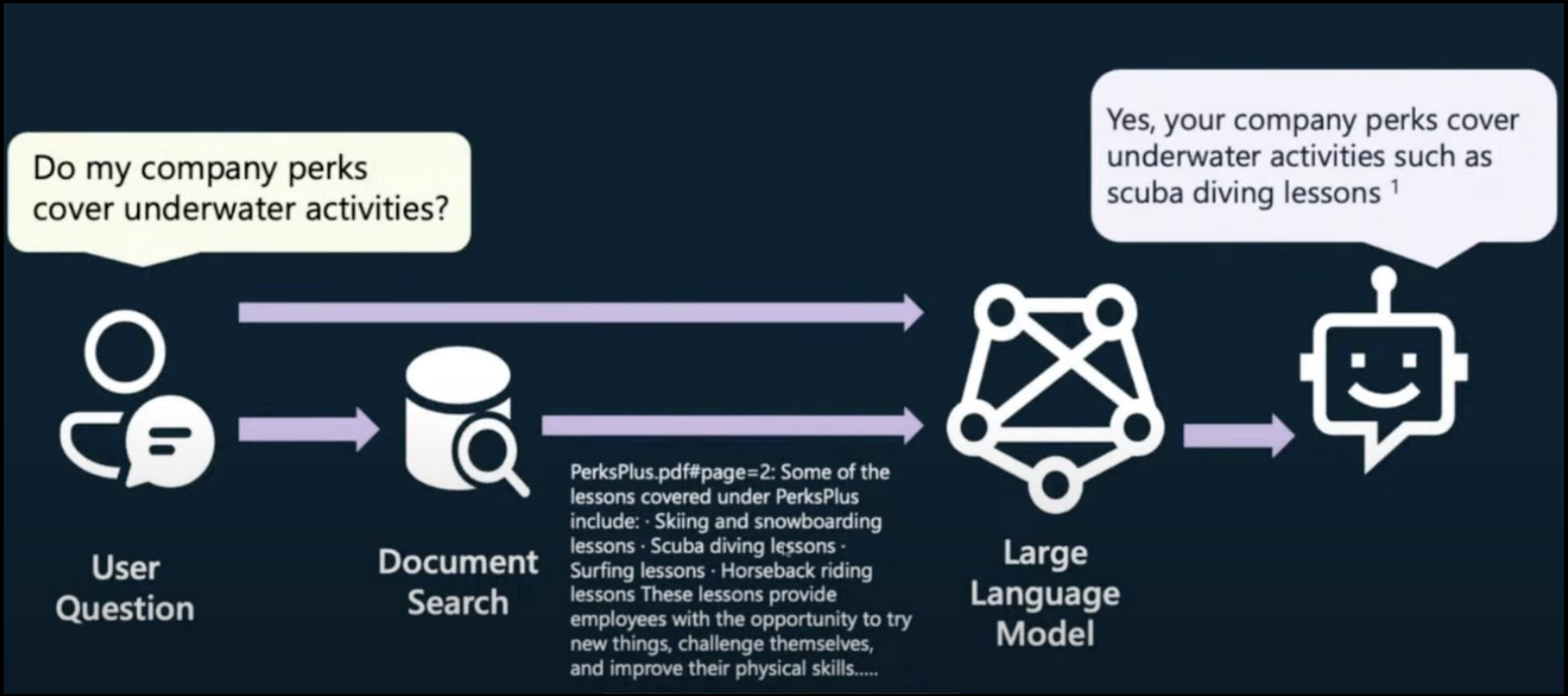
Learn new facts (temporarily)

https://www.youtube.com/watch?v=vuOA13Y_Qzk

The Benefits of RAG

- Up-to-date public knowledge (AZ OpenAI documentation)
- Access to internal knowledge (Company HR docs)

RAG – Retrieval Augmented Generation

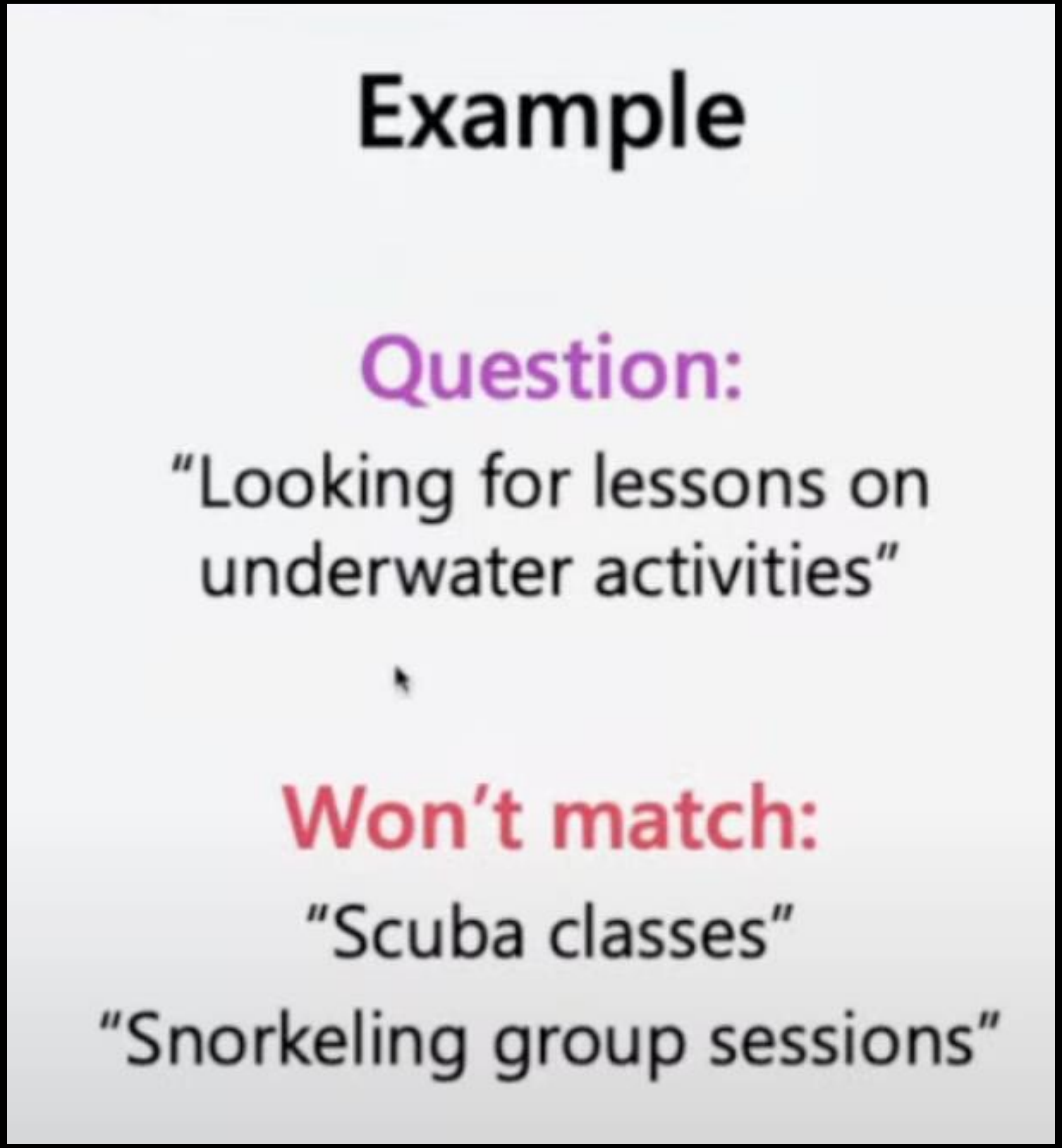

Do my company perks cover underwater activities?
Yes, your company perks cover underwater activities such as scuba diving lessons 1
User Question
Document Search
Large Language Model

PerksPlus.pdf#page=2: Some of the lessons covered under PerksPlus include: · Skiing and snowboarding lessons · Scuba diving lessons · Surfing lessons · Horseback riding lessons These lessons provide employees with the opportunity to try new things, challenge themselves, and improve their physical skills.....

https://www.youtube.com/watch?v=vuOA13Y_Qzk

Robust retrieval for RAG

- Responses only as good as retrieved data
- Keyword search recall challenges
- Vector-based retrieval finds docs by Semantic similarity


Example

Question:
"Looking for lessons on underwater activities"

Won't match:
"Scuba classes"
"Snorkeling group sessions"

Vector embeddings

- An embedding encodes an input as a list of FP numbers
- “dog” -> [0.014, -0.05, ...]
- Different models output different embeddings (different lengths)

https://aka.ms/aitour/vectors

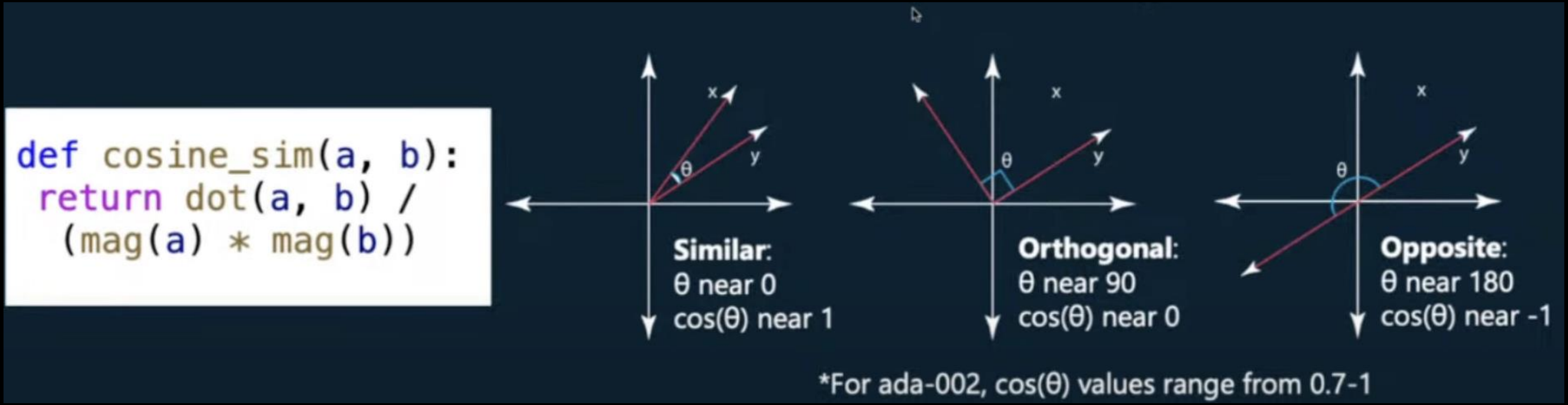
https://pamelafox.github.io/vectors-comparison/

https://pamelafox.github.io/vectors-comparison/movies.html

https://github.com/Azure-Samples/rag-with-azure-ai-search-notebooks/blob/main/vector_embeddings.ipynb

Vector similarity

Embeddings are used to calculate similarity between inputs:
The most common distance measurement is cosine similarity



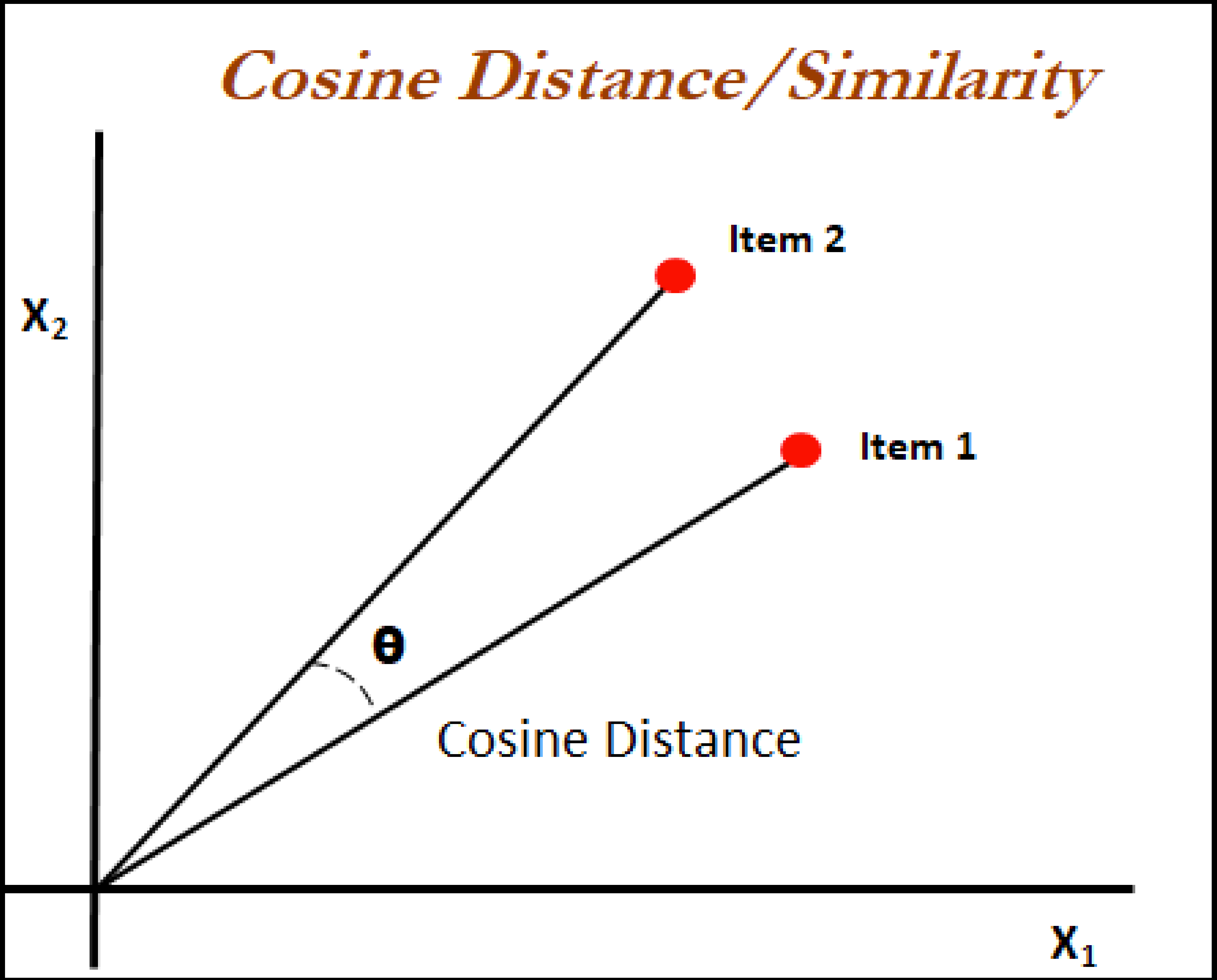
```
def cosine_sim(a, b):
  return dot(a, b) /
   (mag(a) * mag(b))
```

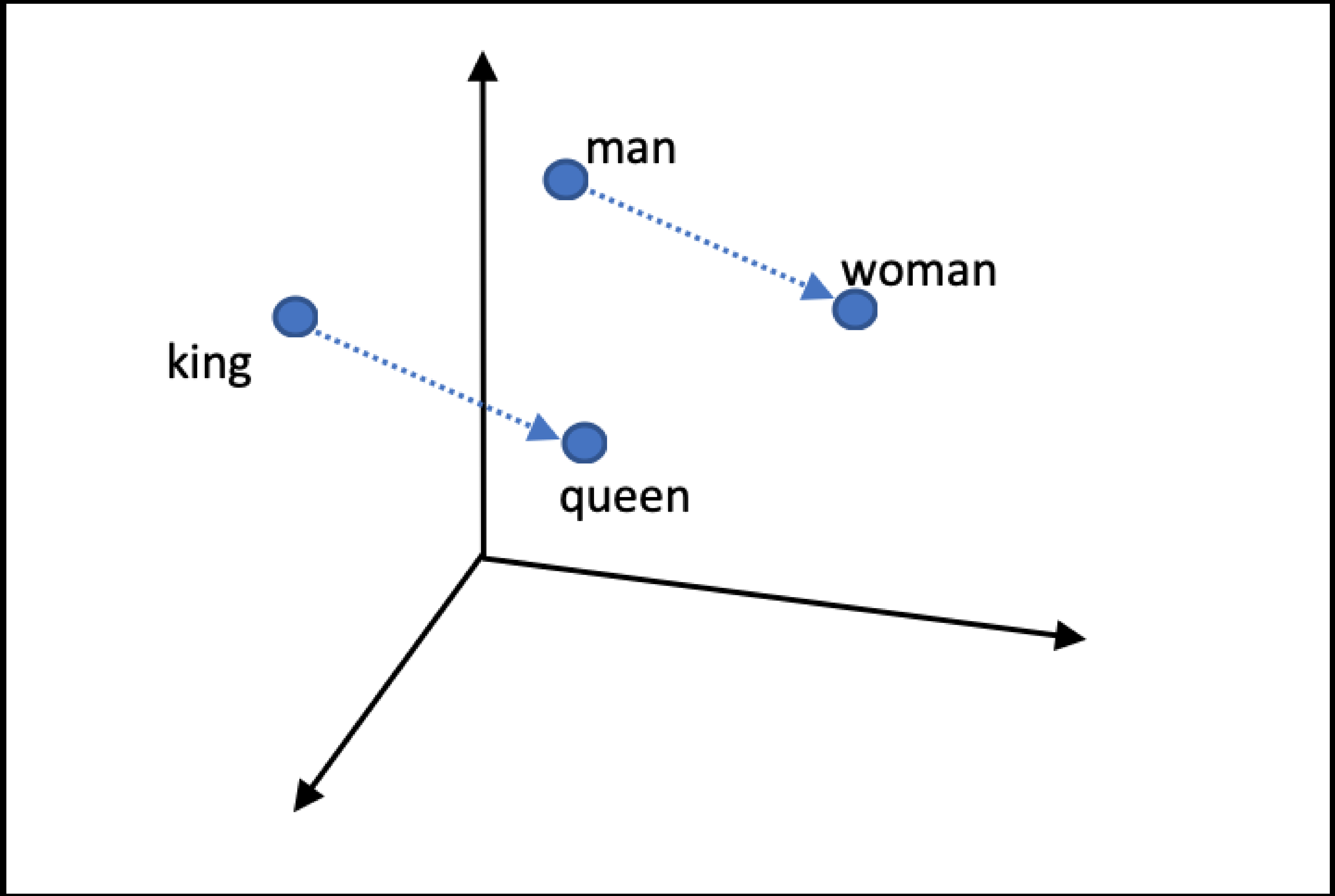
*For ada-002, cos(θ) values range from 0.7-1

https://aka.ms/aitour/vectors

https://github.com/Azure-Samples/rag-with-azure-ai-search-notebooks/blob/main/vector_embeddings.ipynb

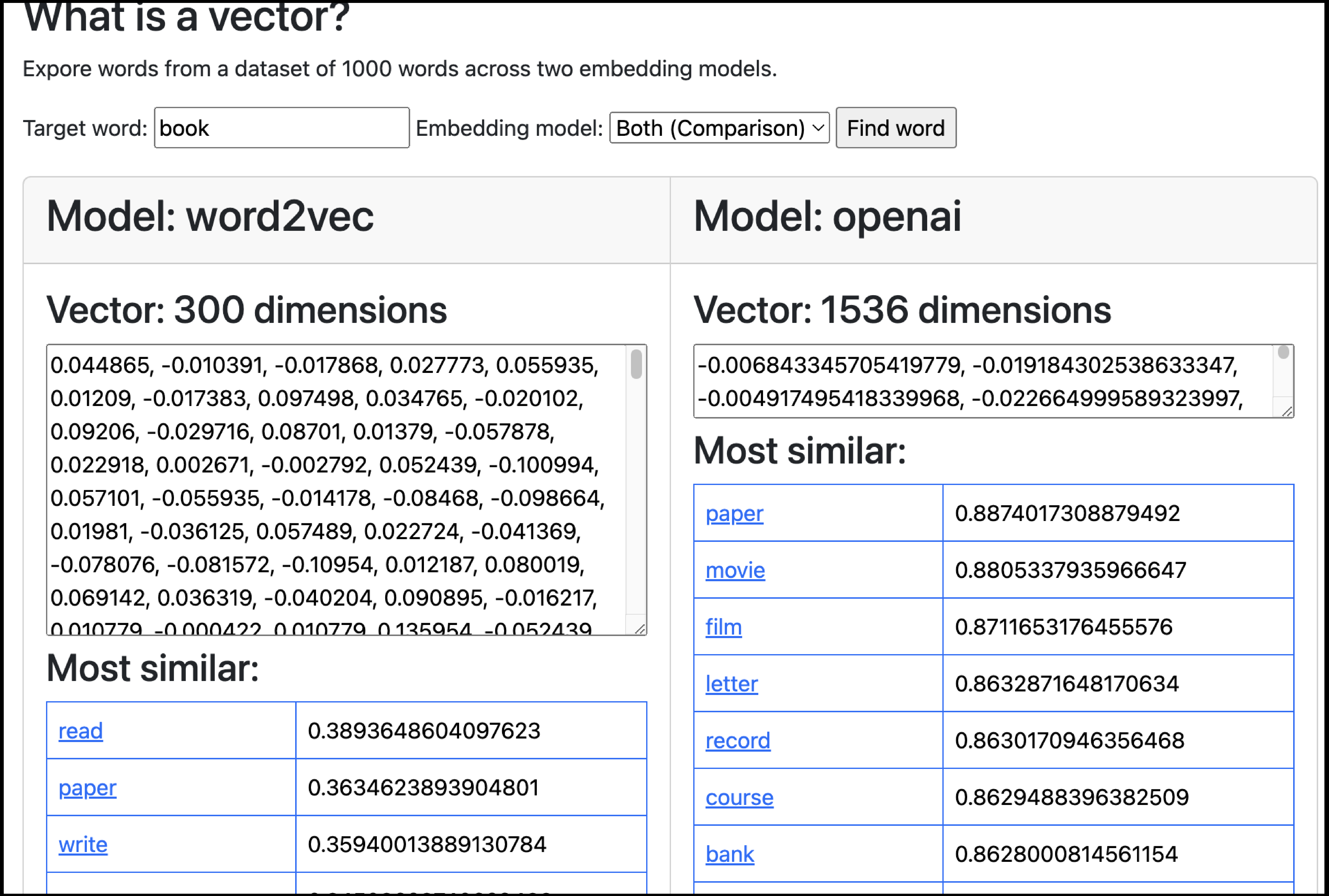
Vector embeddings





https://github.com/Azure-Samples/rag-with-azure-ai-search-notebooks/blob/main/vector_embeddings.ipynb

Vector Comparison



What is a vector?

Expore words from a dataset of 1000 words across two embedding models.

Target word: book Embedding model: Both (Comparison) Find word

Model: word2vec

Vector: 300 dimensions

```
0.044865, -0.010391, -0.017868, 0.027773, 0.055935,
0.01209, -0.017383, 0.097498, 0.034765, -0.020102,
0.09206, -0.029716, 0.08701, 0.01379, -0.057878,
0.022918, 0.002671, -0.002792, 0.052439, -0.100994,
0.057101, -0.055935, -0.014178, -0.08468, -0.098664,
0.01981, -0.036125, 0.057489, 0.022724, -0.041369,
-0.078076, -0.081572, -0.10954, 0.012187, 0.080019,
0.069142, 0.036319, -0.040204, 0.090895, -0.016217,
0.010779, -0.000422, 0.010779, 0.135954, -0.052439
```

Most similar:

read	0.3893648604097623
paper	0.3634623893904801
write	0.35940013889130784

Model: openai

Vector: 1536 dimensions

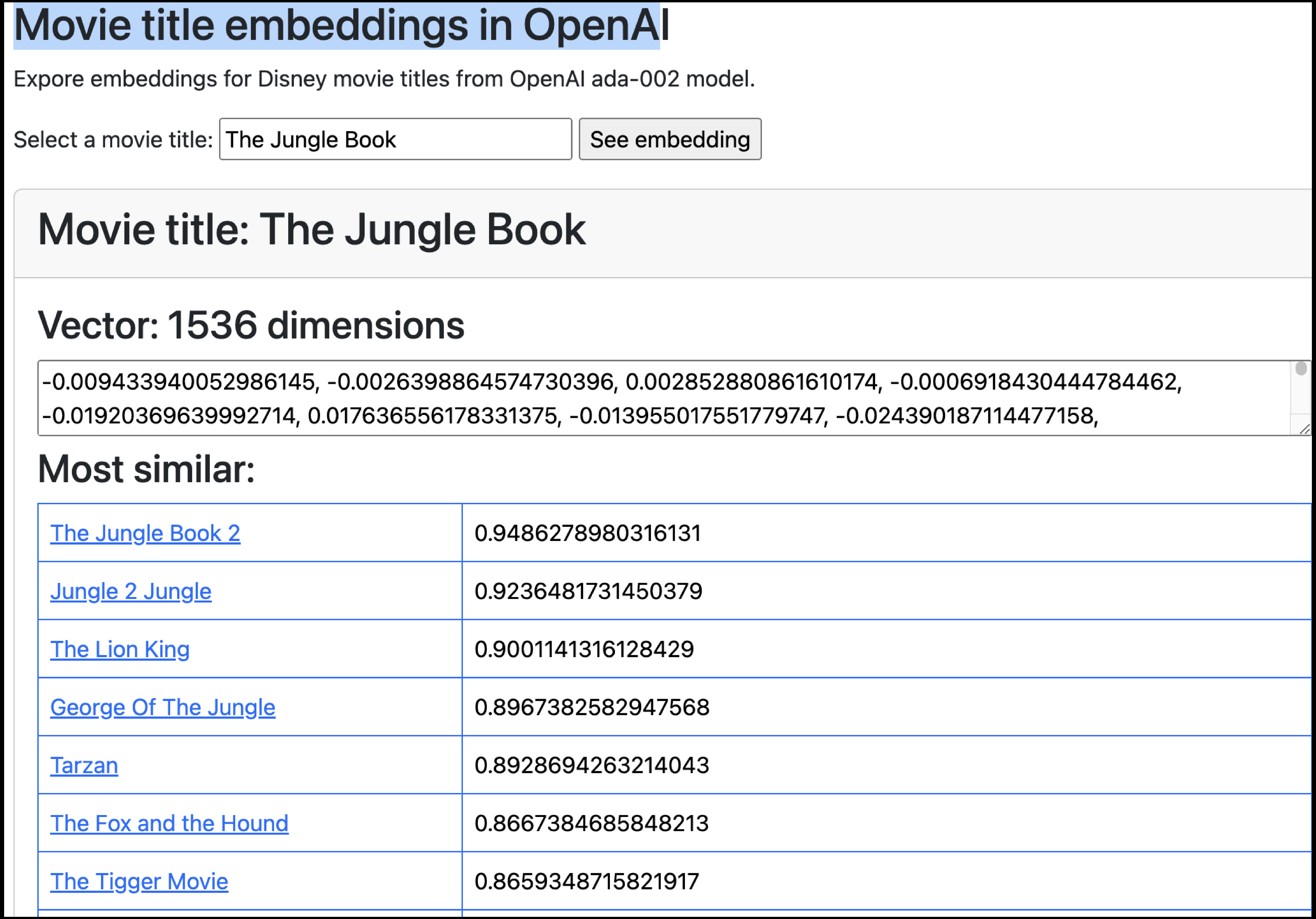
```
-0.006843345705419779, -0.019184302538633347,
-0.004917495418339968, -0.022664999589323997,
```

Most similar:

paper	0.8874017308879492
movie	0.8805337935966647
film	0.8711653176455576
letter	0.8632871648170634
record	0.8630170946356468
course	0.8629488396382509
bank	0.8628000814561154

https://pamelafox.github.io/vectors-comparison/

Movie title embeddings in OpenAI



Movie title embeddings in OpenAI

Expore embeddings for Disney movie titles from OpenAI ada-002 model.

Select a movie title: The Jungle Book [See embedding]

Movie title: The Jungle Book

Vector: 1536 dimensions

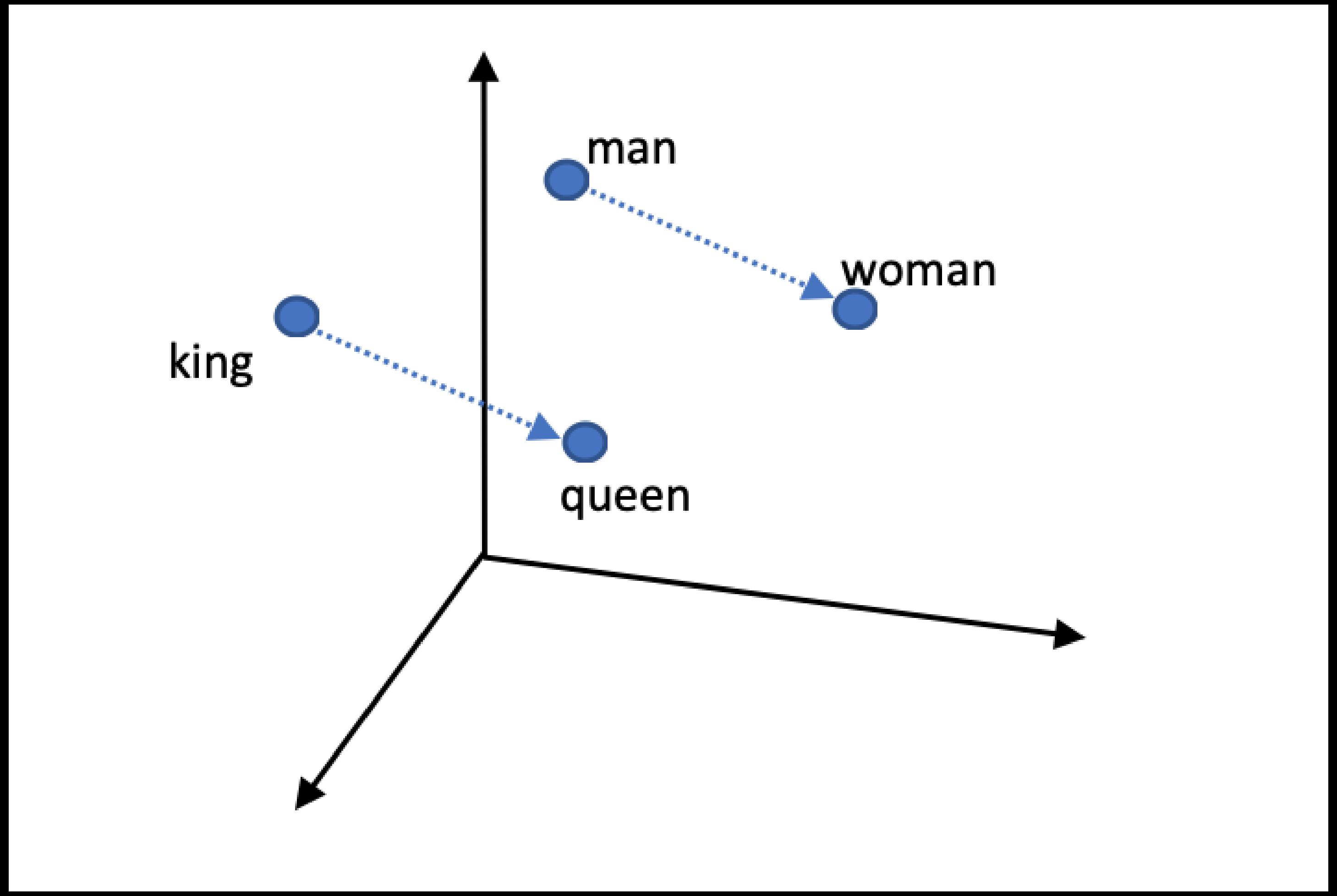
-0.009433940052986145, -0.0026398864574730396, 0.002852880861610174, -0.0006918430444784462,
-0.01920369639992714, 0.017636556178331375, -0.013955017551779747, -0.024390187114477158,

Most similar:

The Jungle Book 2	0.9486278980316131
Jungle 2 Jungle	0.9236481731450379
The Lion King	0.9001141316128429
George Of The Jungle	0.8967382582947568
Tarzan	0.8928694263214043
The Fox and the Hound	0.8667384685848213
The Tigger Movie	0.8659348715821917

https://pamelafox.github.io/vectors-comparison/movies.html

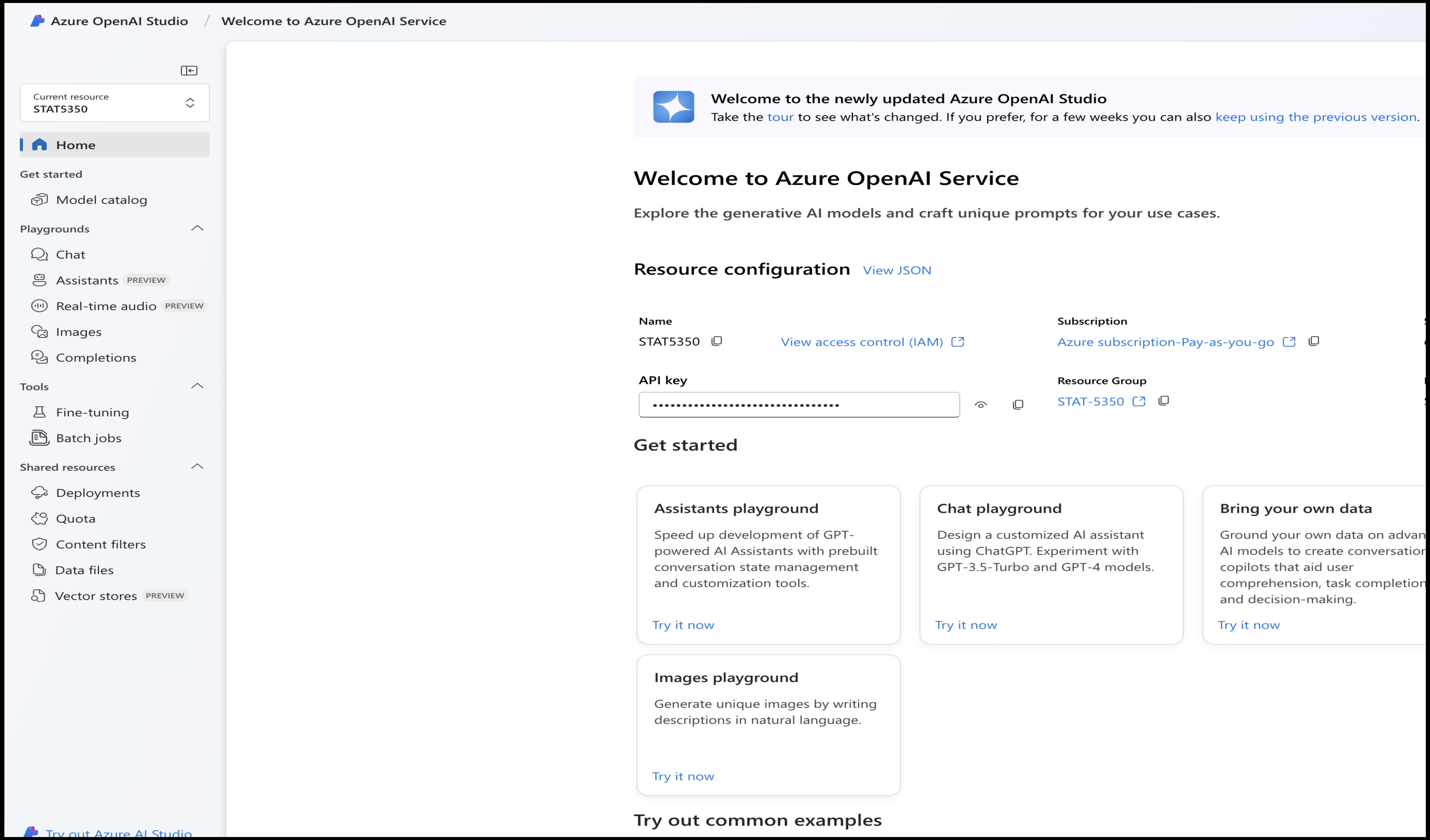
Vector embeddings Lab



https://aka.ms/aitour/vectors

https://github.com/Azure-Samples/rag-with-azure-ai-search-notebooks/blob/main/vector_embeddings.ipynb

Azure OpenAI



Azure OpenAI Studio / Welcome to Azure OpenAI Service

Current resource
STAT5350

Welcome to the newly updated Azure OpenAI Studio
Take the tour to see what's changed. If you prefer, for a few weeks you can also keep using the previous version.

Welcome to Azure OpenAI Service

Explore the generative AI models and craft unique prompts for your use cases.

Resource configuration View JSON

Name
STAT5350 View access control (IAM)

Subscription
Azure subscription-Pay-as-you-go

API key
••••••••••••••••••••••••••••••••

Resource Group
STAT-5350

Get started

Assistants playground
Speed up development of GPT-powered AI Assistants with prebuilt conversation state management and customization tools.
Try it now

Chat playground
Design a customized AI assistant using ChatGPT. Experiment with GPT-3.5-Turbo and GPT-4 models.
Try it now

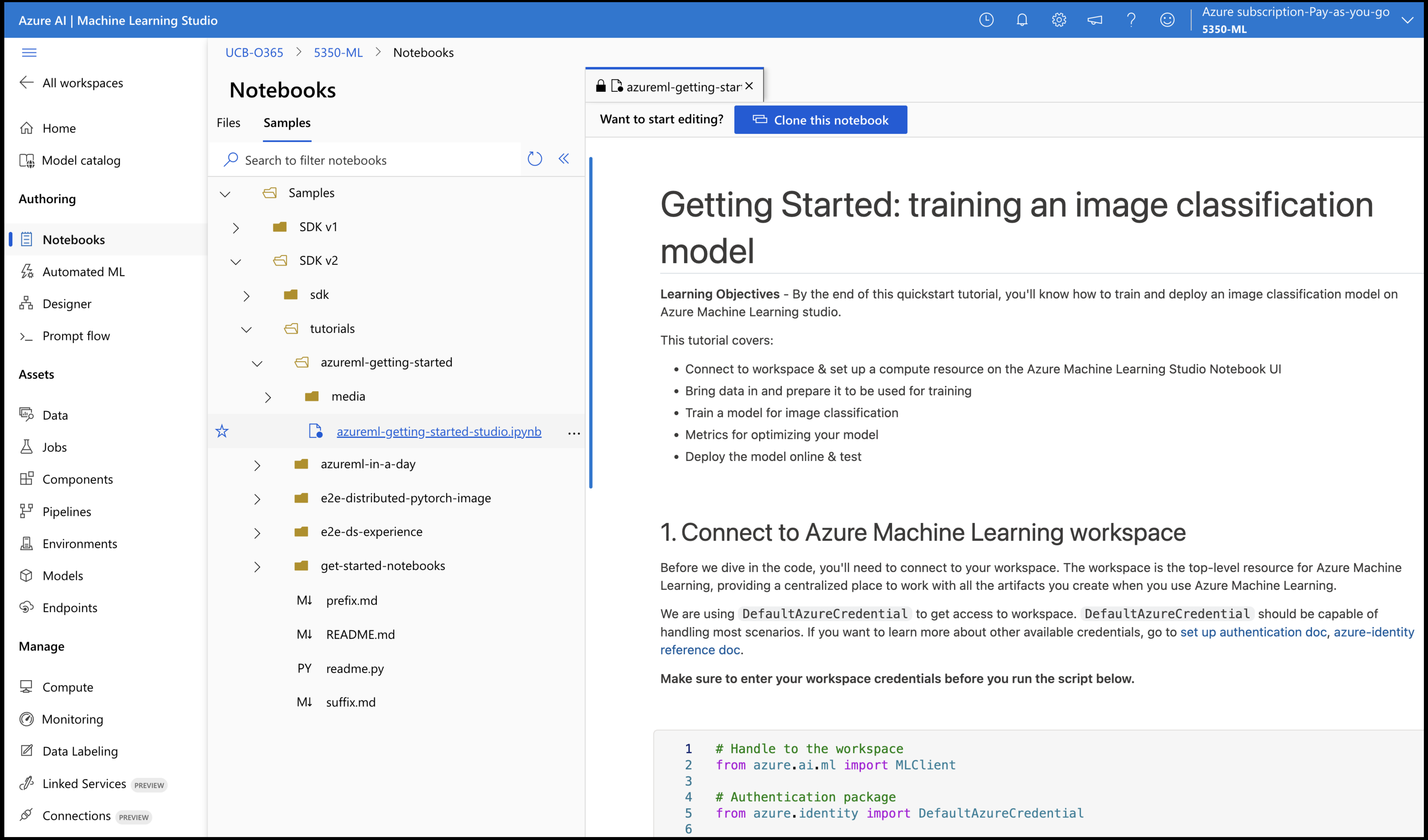
Bring your own data
Ground your own data on advan
AI models to create conversation
copilots that aid user
comprehension, task completion
and decision-making.
Try it now

Images playground
Generate unique images by writing descriptions in natural language.
Try it now

Try out common examples

https://learn.microsoft.com/en-us/azure/ai-services/openai/

Azure ML



Getting Started: training an image classification model

Learning Objectives - By the end of this quickstart tutorial, you'll know how to train and deploy an image classification model on Azure Machine Learning studio.

This tutorial covers:

- Connect to workspace & set up a compute resource on the Azure Machine Learning Studio Notebook UI
- Bring data in and prepare it to be used for training
- Train a model for image classification
- Metrics for optimizing your model
- Deploy the model online & test

1. Connect to Azure Machine Learning workspace

Before we dive in the code, you'll need to connect to your workspace. The workspace is the top-level resource for Azure Machine Learning, providing a centralized place to work with all the artifacts you create when you use Azure Machine Learning.

We are using `DefaultAzureCredential` to get access to workspace. `DefaultAzureCredential` should be capable of handling most scenarios. If you want to learn more about other available credentials, go to set up authentication doc, azure-identity reference doc.

Make sure to enter your workspace credentials before you run the script below.

```
# Handle to the workspace
from azure.ai.ml import MLClient

# Authentication package
from azure.identity import DefaultAzureCredential

```

https://learn.microsoft.com/en-us/azure/machine-learning/tutorial-explore-data?view=azureml-api-2

- Azure OpenAI
- RAG
- Exercise

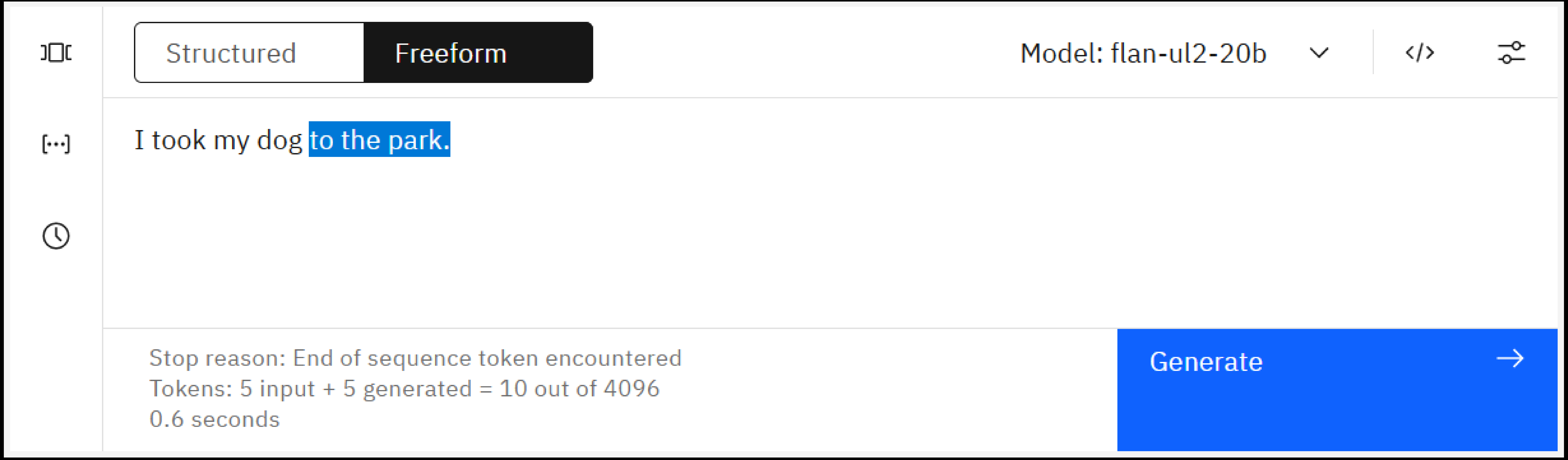
Prompt Tips

https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/fm-prompt-tips.html?context=wx

Tip 1: Remember that everything is text completion

Your *prompt* is the text you submit for processing by a foundation model.
For most models, simply asking a question or typing an instruction won't yield the best results. That's because the model isn't *answering* your prompt, the model is *appending text to it*.

This image demonstrates prompt text and generated output:
•Prompt text: "I took my dog "
•Generated output: "to the park."



I took my dog to the park.

Stop reason: End of sequence token encountered
Tokens: 5 input + 5 generated = 10 out of 4096
0.6 seconds

Tip 2: Include all the needed prompt components

Effective prompts usually have one or more of the following components: instruction, context, examples, and cue.

Instruction

An instruction is an imperative statement that tells the model what to do. For example, if you want the model to list ideas for a dog-walking business, your instruction could be: "List ideas for starting a dog-walking business:”

Context

Including background or contextual information in your prompt can nudge the model output in the right direction. Specifically, words that appear in your prompt text are more likely to be included in the generated output.

Examples

To indicate the format or shape that you want the model response to be, include one or more pairs of example input and corresponding output that illustrate the pattern you want the generated text to follow.
Including one example in your prompt is called *one-shot prompting*, including two or more examples in your prompt is called *few-shot* prompting, and when your prompt has no examples, that's called *zero-shot* prompting.
When you prompt models that have been tuned, you might not need to include examples.

Cue

A cue is text at the end of the prompt that is likely to start the generated output on the right path. Remember, as much as it seems like the model is *responding to your prompt*, the model is really *appending text to your prompt* or *continuing your prompt*.

Tip 3: Include descriptive details

The more guidance, the better. Experiment with including descriptive phrases related to aspects of your ideal result: content, style, and length. Including these details in your prompt can cause a more creative or more complete result to be generated.

For example, you can improve the sample instruction by adding descriptive information as follows:

•Original: "List ideas for starting a dog-walking business"
•Improved: "List ideas for starting a large, wildly successful dog-walking business”

Example:

Before
A prompt with a simple instruction:
List ideas for starting a dog-walking business.
The resulting foundation model output isn't great:
Take care of the dog and walk it in your neighborhood. You can create flyers and put them around the neighborhood, you can also put up ads on websites where dog lovers hang out.

After
A prompt with all of the recommended prompt components:

Tip 3: Include descriptive details

Instruction:
Based on the following industry research, suggest ideas for starting a large, wildly successful dog-walking business.

Context:
Industry research:
***
The most successful dog-walking businesses cater to owners' needs and desires while also providing great care to the dogs. For example, owners want flexible hours, a shuttle to pick up and drop off dogs at home, and personalized services, such as custom meal and exercise plans. Consider too how social media has permeated our lives. Web-enabled interaction provide images and video that owners will love to share online, which is great advertising for the business.
***

Tip 3: Include descriptive details

Example
Ideas for starting a lemonade business:
- Set up a lemonade stand
- Partner with a restaurant
- Get a celebrity to endorse the lemonade

Cue:
Ideas for starting a large, wildly successful dog-walking business:

RESPONSE:
•*Start by doing some basic market research. Interview dog owners in your area and ask them what they would want from their dog walker.*
•*Offer a free sample or trial period - like several weeks - so you can see if they are really interested.*
•*Give personalized service. Offer to train the owner's dog to get in a proper sit or down for the dog walker. Train the human too!*
•*Be friendly and outgoing. Social media is huge and people love to post photos of themselves with their pets.*
•*Use technology to help keep costs low, such as using web-enabled interactions. For example, you could offer a coupon on a popular pet food store website.*

Prompting Principles

Principle 1

Write clear and specific instructions

Tactics:

- *Tactic 1 – Use Delimiters to clearly indicate distinct parts of the input*
- *Tactic 2 – Ask for structured output*
- *Tactic 3 – Ask the model to check whether conditions are satisfied*
- *Tactic 4 – Try ‘few—shot’ prompting*

Prompting Principles

Principle 2

Give the model time to think

Tactics:

- *Tactic 1 – Specify the steps require to complete a task*
- *Tactic 2 – Instruct the model to work out its own solution before rushing to conclusion*
- *Tactic 3 – Watch out for hallucinations*

RAG Lab