Building AI Agents / Copilots

Dani Diaz

Principal Technical Architect

Global Black Belt – Microsoft AI

LLMs are like Language Calculator

Prompt:
A realistic photo of a less futuristic language calculator on a desk in a classroom with a pencil and a spiral notebook

Language Calculator

Will my sleeping bag work for my trip to Patagonia next month?

+

Historical weather lookup

Intent Mapping

Personalization

Context

Recommendations Engine

???

=

Yes, your Elite Eco sleeping bag is rated to 21.6F, which is below the average low temperature in Patagonia in November

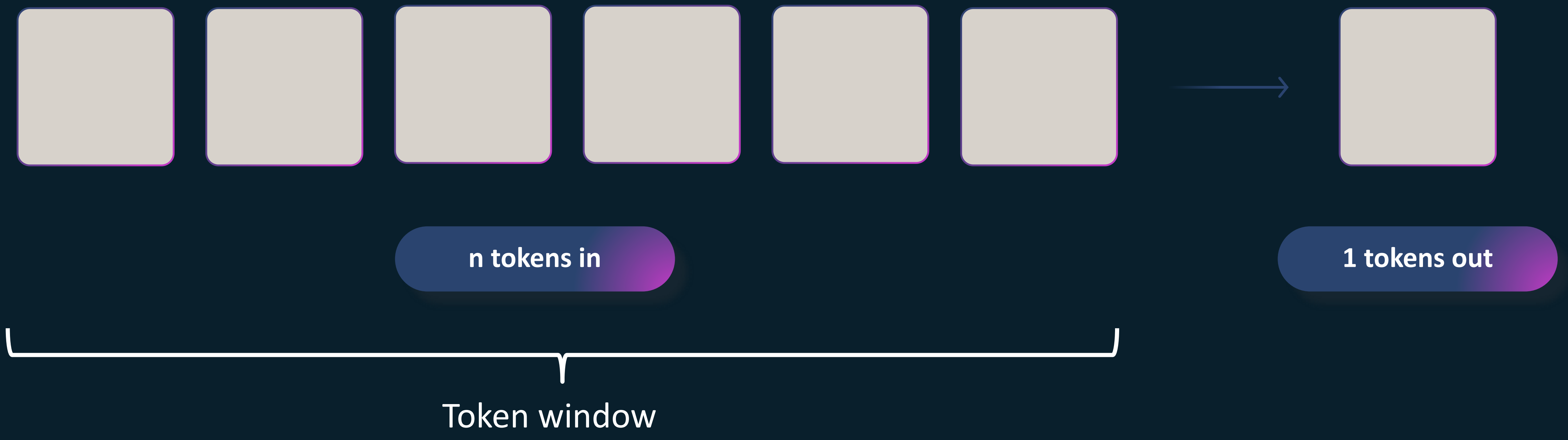
User input

Prompt engineering

LLM

Output

How language models work



How language models work

Tokens

Tokens	Characters
11	43

We need to stop anthropomorphizing ChatGPT.

https://platform.openai.com/tokenizer

How language models work

in out

We need to stop

We need to stop anthrop

We need to stop anthropomorph

We need to stop anthropomorphizing

We need to stop anthropomorphizing Chat

We need to stop anthropomorphizing ChatG

We need to stop anthropomorphizing ChatGPT

We need to stop anthropomorphizing ChatGPT.

Token window

What is Prompt Engineering?

- Prompting: the instruction of an AI model to do a task
  - A prompt can be a simple or complex
- Examples of simple prompts:
  - A yes/no question
  - A fact-based question (questions are often called ‘standard’ prompts)
  - A mathematical equation (LMs aren’t optimized for math so can get it incorrect)
  - Summarize or paraphrase a piece of text
- **Prompt engineering** is an NLP concept that involves discovering prompts that yield desirable or useful results.
  - How do we ask a question that will give us a better answer?
  - How do we give more context to help guide the model without retraining or fine-tuning?
  - More of an art than a science

Why is Prompt Engineering important?

- GPT models are *generative* LMs: AI that generates output
  - *Generative:* more difficult to predict how it will respond, even when given the same prompt
  - vs *Discriminative AI:* typically produce the same output when given the same input.
- Earlier Machine Learning (ML) and LMs were typically fine-tuned for specific tasks e.g., training on additional email data for email reply prediction
- GPT models have excellent generalized knowledge of language
  - Not fine-tuned to specific types of tasks
  - GPT models can still be fine-tuned, but this is complex and expensive
- However, we can improve the relevancy and accuracy of responses with better prompts and providing intelligent and well-thought-out relevant content
  - Think of it as on-the-fly fine-tuning

Basics of Prompting

- A prompt contains any of the following elements:
  - **Instruction** - a specific task or instruction you want the model to perform
  - **Context** - external information or additional context that can steer the model to better responses
  - **Input Data** - the input or question that we are interested to find a response for
  - **Output Indicator** - the type or format of the output.
  - You do not need all the four elements for a prompt and the format depends on the task at hand. We will touch on more concrete examples in upcoming guides.

Prompt Engineering

Prompt

Write a tagline for an ice cream shop.

Response

We serve up smiles with every scoop!

Zero Shot Prompt

Classify the following url:

www.espn.com

Response

Sports

Few Shot Prompt

Given the following examples of web pages and their classes:

Page: www.espn.com Class: Sports
Page: www.google.com Class: Search

Classify the following page:

Page: www.allrecipes.com Class:

Response

Cooking

RAG

You know the following:
[vectors]

Answer the following question:

How many US paid holidays are there?

Response

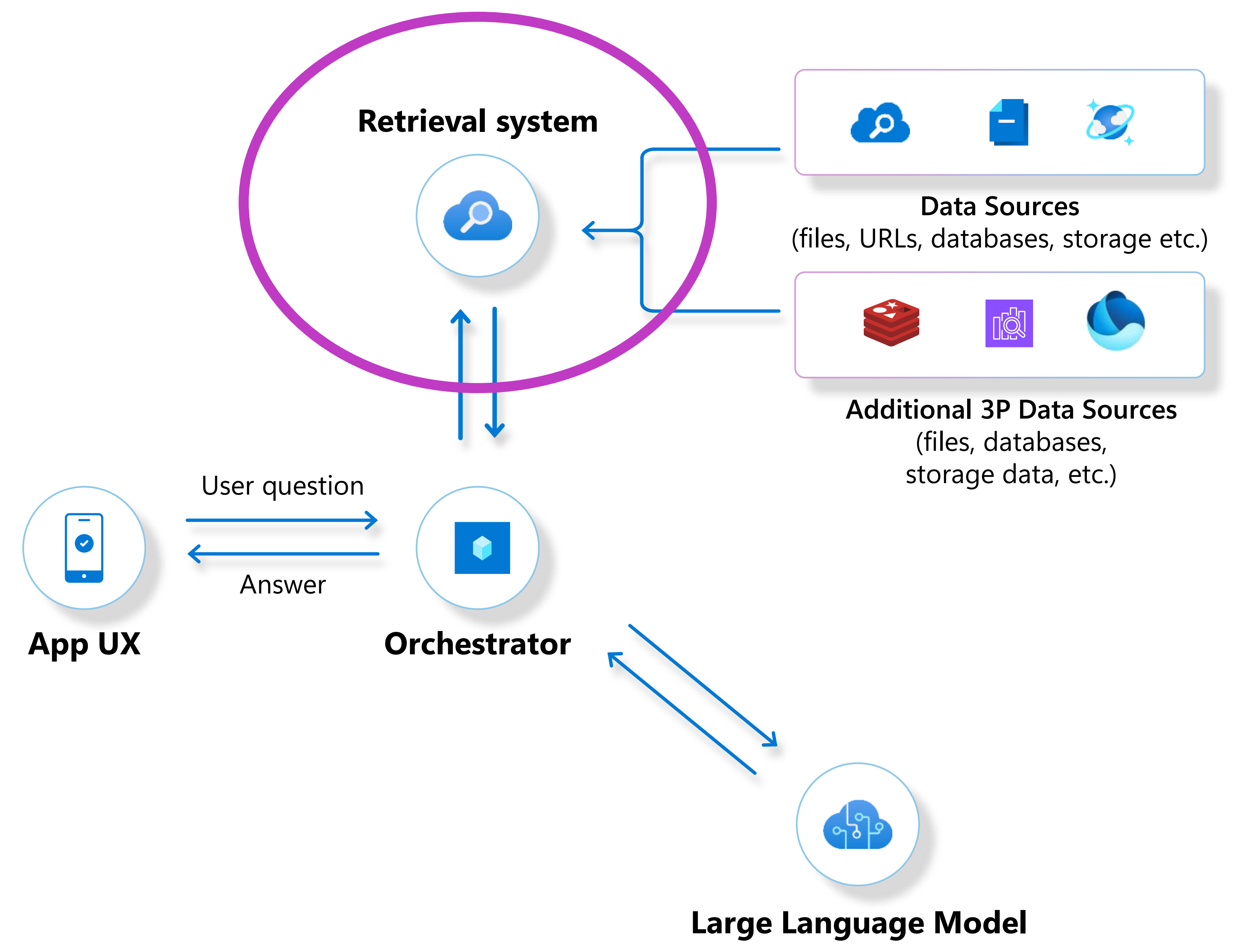
There are 10 paid holidays in the US.

Hallucination

- Tell the model what you don’t want
- Tell it what to say when it is not sure, say “I don’t know”
- “Do not make up facts”
- Dynamically finding and injecting relevant context into prompt
- Discriminator that checks if all information needed to answer is available
- Step by step reasoning
- Ask the model to explain along with the answer

Azure AI Search

Retrieval Augmented Generation



Bring domain knowledge to LLMs

Prompt engineering

In-context learning

Fine tuning

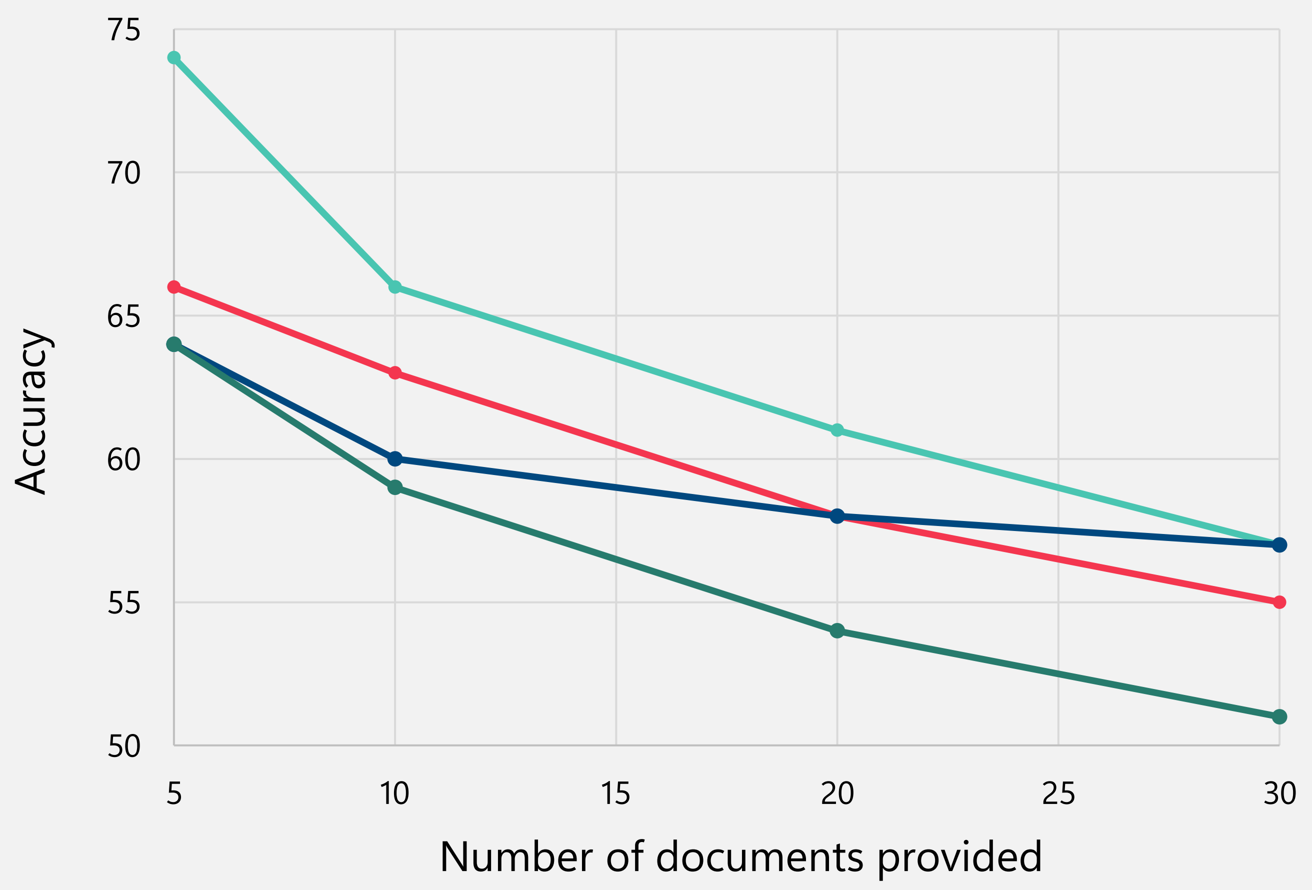
Learn new skills

Retrieval augmentation

Learn new facts

Your retrieval strategy matters

More information ≠ better results



Source: Lost in the Middle: How Language Models Use Long Contexts, Liu et al. arXiv:2307.03172

Robust retrieval for RAG apps

- Responses only as good as retrieved data
- Keyword search recall challenges
  - "Vocabulary gap"
  - Gets worse with natural language questions
- Vector-based retrieval finds documents by semantic similarity
  - Robust to variation in how concepts are articulated (word choices, morphology, specificity, etc.)

Question:

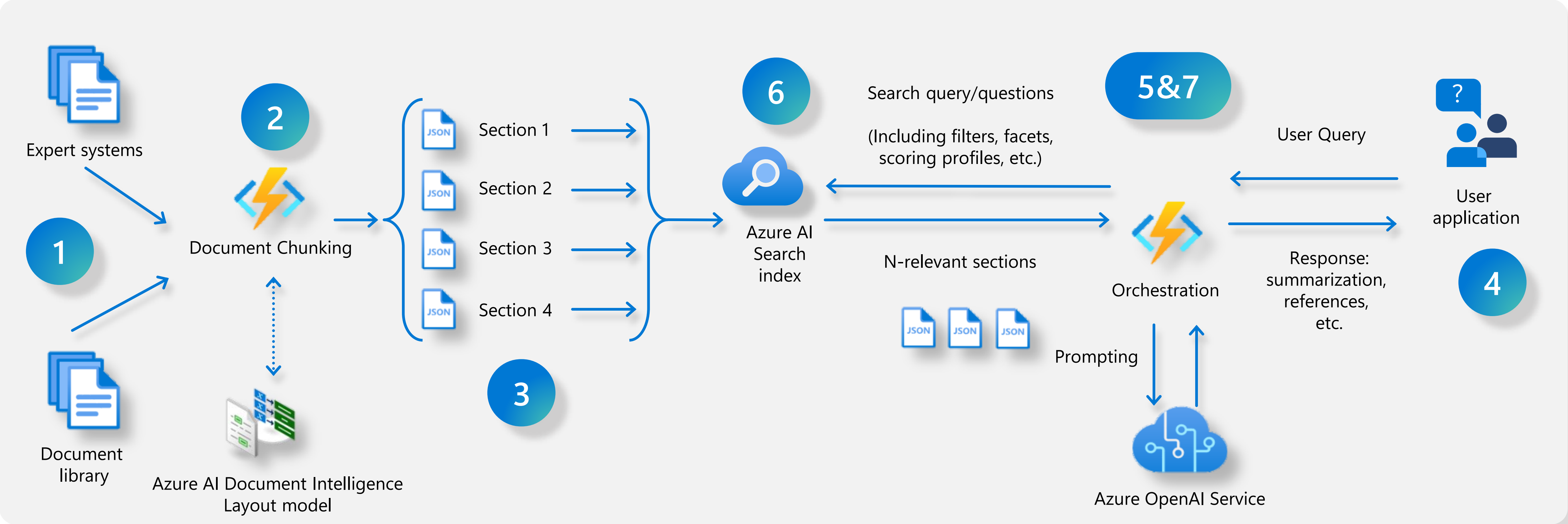
"Does my **health plan** cover **annual eye** exams?"

Won't match:

"Northwind Standard only offers coverage for vision exams and glasses."

"Northwind Health Plus offers coverage for vision exams, glasses, and contact lenses, as well as dental exams, cleanings, and fillings."

Anatomy of RAG



1. Data ingestion

Different data formats and system of records

2. Chunking

What is the best Chunking strategy?

3. Indexing

Shall I use vector embeddings data transformation, mappings?

4. User interface

Chatbot for Q&A surfaced to end users

5. Orchestration

Communication coordination and prompting—Prompt to get retriever query

6. Data retrieving

Shall I use vector, semantic, keyword or hybrid approach?

7. Orchestration

Communication coordination: create user response based on retrieve data and send to User app

Your document Chunking strategy matters

Chunking solves 3 problems for generative AI applications:

1. Allows multiple retrieved documents to be passed to the LLM within its context window limit
2. Provides a mechanism for the most relevant passages of a given document to be ranked first
3. Vector search has a per-model limit to how much content can be embedded into each vector

Retrieval Configuration	Single vector per document *[Recall@50]*	Chunked documents *[Recall@50]*
Queries whose answer is in long documents	28.2	**45.7**
Queries whose answer is deep into a document	28.7	**51.4**

Chunk boundary strategy	Recall@50
512 tokens, break at token boundary	40.9
512 tokens, preserve sentence boundaries	42.4
512 tokens with 10% overlapping chunks	43.1
512 tokens with 25% overlapping chunks	**43.9**

Retrieval Configuration	Recall@50
512 input tokens per vector	**42.4**
1024 input tokens per vector	37.5
4096 input tokens per vector	36.4
8191 input tokens per vector	34.9

Chunking strategies

Chunking methods

Method	Description	
Fixed-size Chunking	Simple and common, maintaining a balance between overlap and semantic context	→ AI Search Built-in Split Skill
Specialized Chunking	For structured content like Markdown and LaTeX	→ AI Search Custom Skill
"Content-aware" Chunking	Content splitting using NLP libraries such as AI Document Intelligence Layout API	→ AI Search Custom Skill
Recursive Chunking	Divides text into smaller Chunks in a hierarchical manner	→ AI Search Custom Skill

Determining the best Chunk size

- Preprocessing data for quality
- Selecting a range of potential Chunk sizes considering content nature and embedding model capabilities
- Evaluating performance of each Chunk size

🔗 Tech blog with suggested Chunking sizes depending on scenario

The technology behind Azure AI Search

Retrieval modes

Keyword-based retrieval

- Traditional full-text search method
- Content is broken into terms; uses the BM25 probabilistic model for scoring

Vector-based retrieval

- Text is converted into vector representations
- Uses embedding models, e.g., Azure Open AI text-embedding-ada-002

Hybrid retrieval

- Combines strengths of Keyword and Vector
- Fusion step selects the best results from both methods, using Reciprocal Rank Fusion (RRF)

Semantic ranking

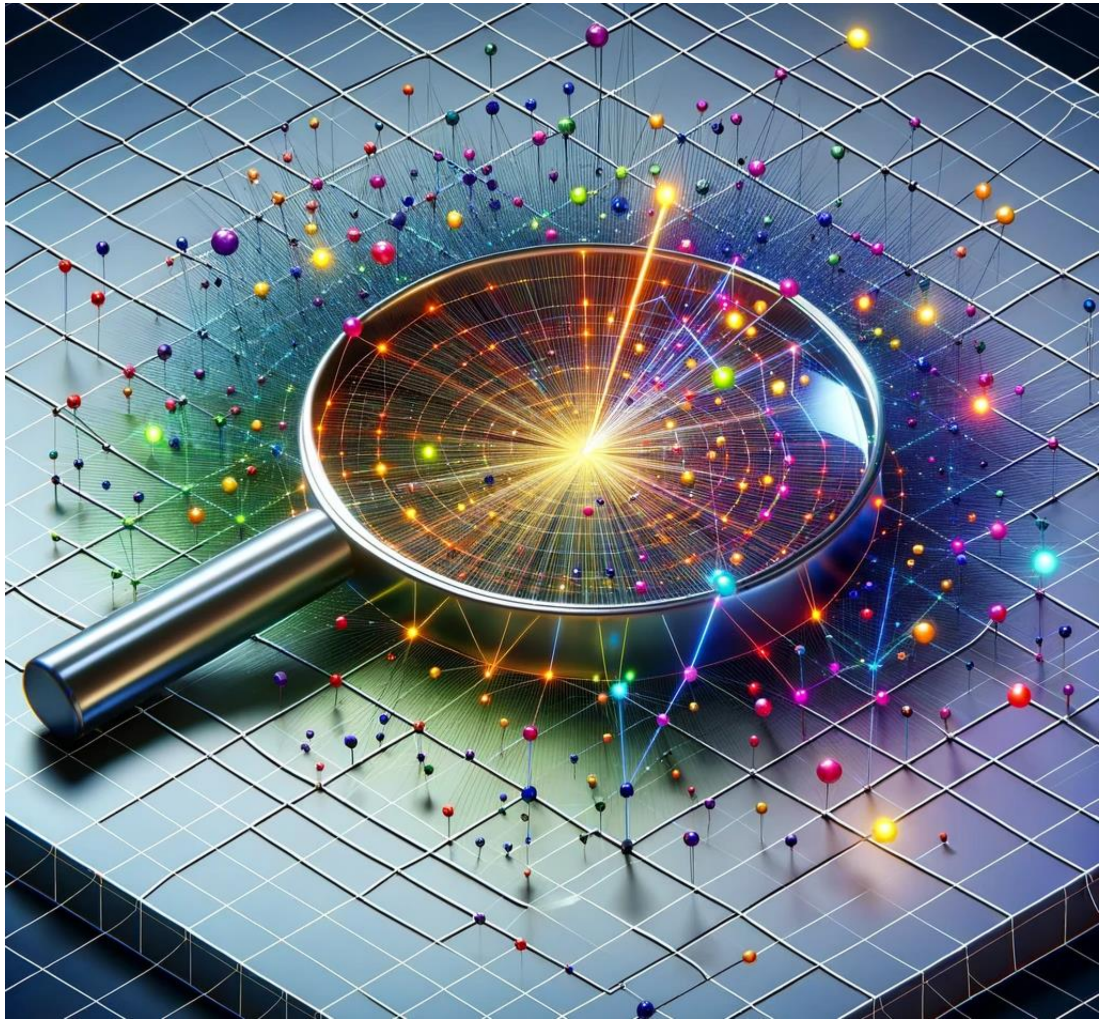
What is Semantic ranking?

- Bing technology that uses transformer models with cross-attention to simultaneously processes query and document text

What does it do?

- Prioritizes the most important results
- Normalized relevance score filters out low-quality results
- Score Range: 0 (irrelevant) to 4 (highly relevant)

Vectors



Learned vector representations

- Models that encode item -> vector
- Similar items map to close vectors
- Sentences, images, graphs, etc.

Vector search

- Find K closest vectors given a “query” vector
- Search exhaustively or through approximations

Vector search

- ✓ Full-featured
- ✓ Create embeddings using any model
- ✓ Explicit and transparent vector data processing
- ✓ Exhaustive KNN search & ANN search
- ✓ Multi-vector

OpenAI Assistant API

At a glance

A tool that helps developers create advanced assistant applications easily. It's designed to be easy to use, with upcoming features that will let it handle more complex tasks. The API simplifies the development process by eliminating the need for multiple integrations to manage state, context windows, and chat threads. It also provides access to powerful native tools and third-party extensibility through function calling.

Effectiveness

A powerful framework that removes the orchestration complexities of building AI solutions

Extensibility

Powerful built in tools with the added capability to seamlessly extend functionality via function calling

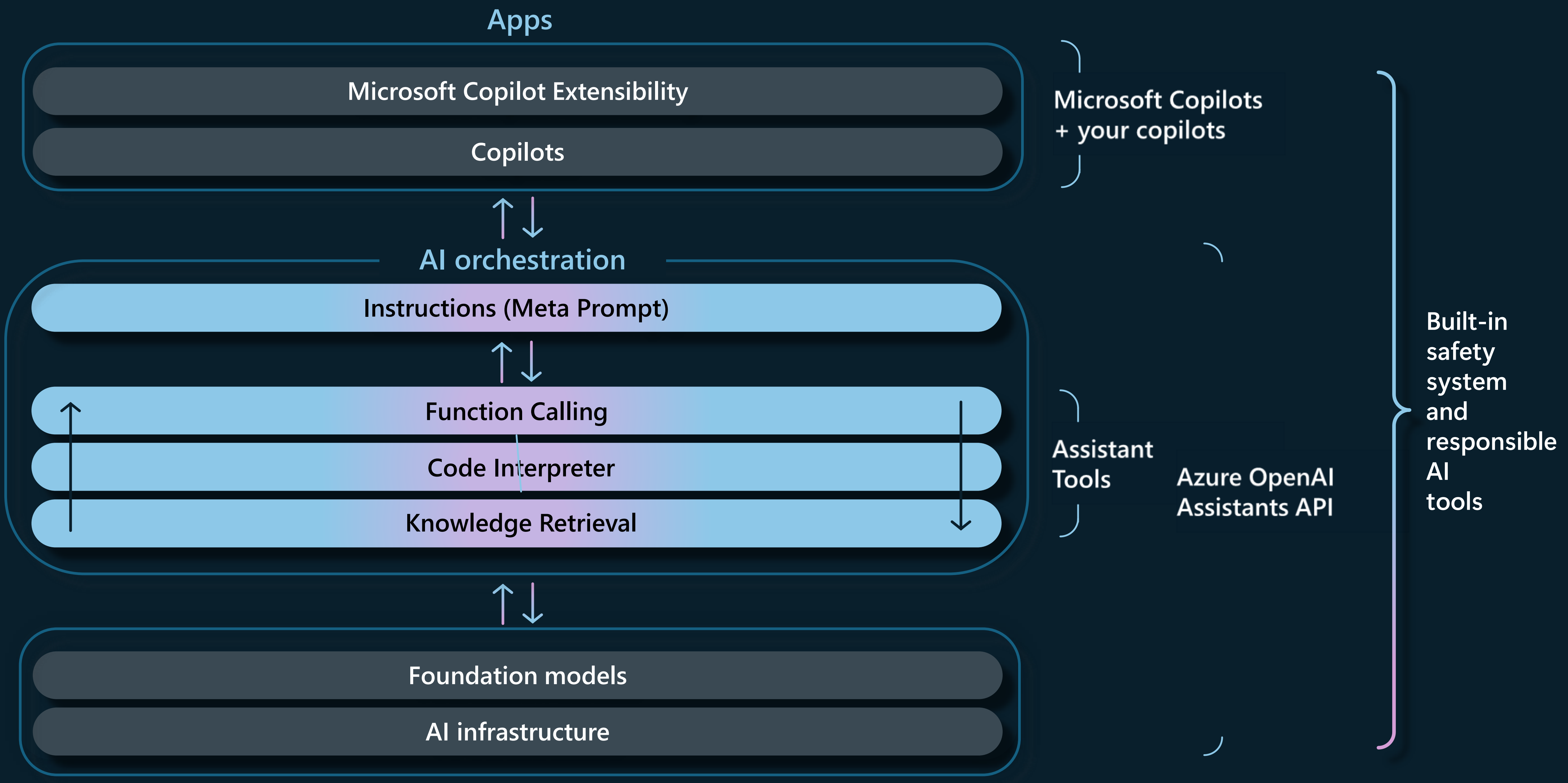
Stateful

Provides a fully stateful experience. No external state mechanism required. Worth the price of admission?

Multi Agent

Envision virtual agent workforces that not only community with each other but also execute tasks to enable end to end flows with the right security guardrails

Assistants Stack



Assistants API

- Can call OpenAI's models with specific instructions to tune their personality and capabilities.
- Can access multiple tools in parallel. These can be both OpenAI-hosted tools — or tools you build / host (via Function calling).
- Can access persistent Threads making it stateful
- Can access files in several formats — either as part of their creation or as part of Threads between Assistants and users. When using tools, Assistants can also create files (e.g., images, spreadsheets, etc) and cite files they reference in the messages they create.

How Assistants Work?

Step 1: Create an Assistant

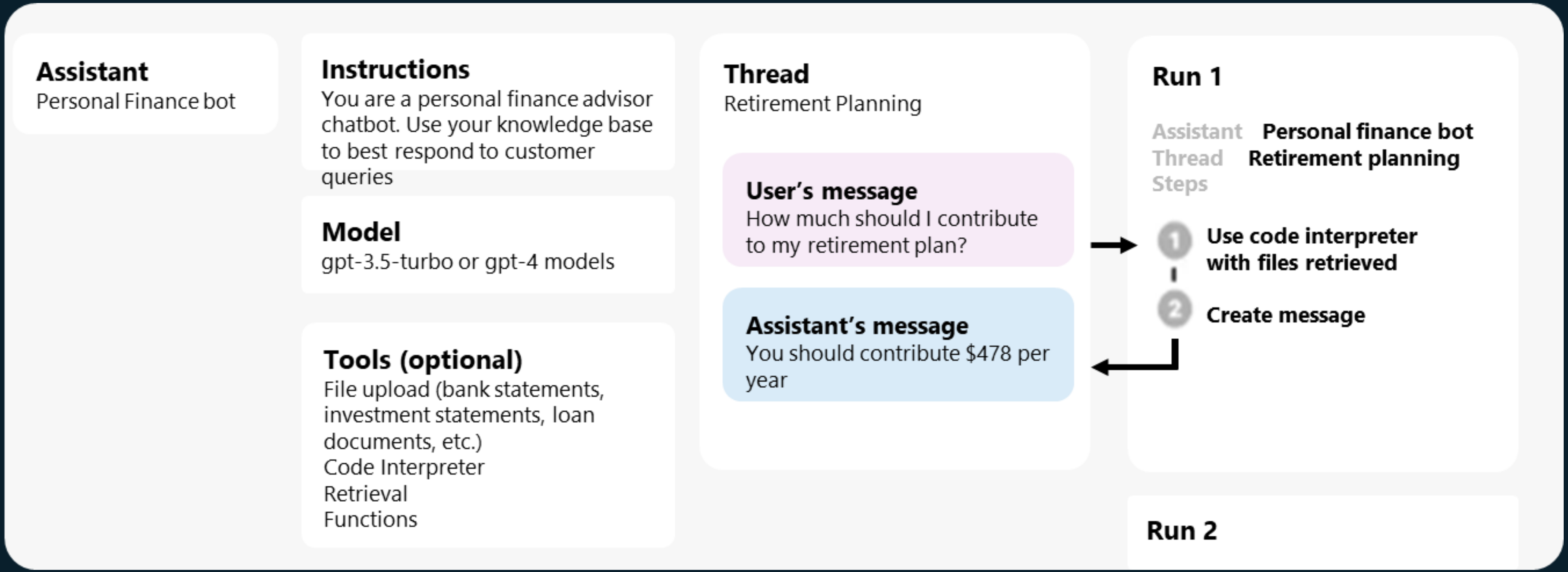
Step 2: Create a Thread

Step 3: Add a Message to a Thread

Step 4: Run the Assistant

Step 5: Check the Run Status

Step 6: Display the Assistant's Response



Objects

Assistant	Purpose-built AI that uses OpenAI's models and calls tools
Thread	A conversation session between an Assistant and a user.
Message	A message created by an Assistant or a user. Messages can include text, images, and other files.
Run	An invocation of an Assistant on a Thread. The Assistant uses its configuration and the Thread's Messages to perform tasks
Run Step	A detailed list of steps the Assistant took as part of a Run. An Assistant can call tools or create Messages during its run. Examining Run Steps allows you to introspect how the Assistant is getting to its final results.

Chat Completions API vs. Assistants API

Chat Completions API

- Lightweight and powerful
- Inherently Stateless

Assistants API

- Stateful (inbuilt conversation state management)
- Access persistent Threads
- Access files in several formats. API handles chunking, embeddings storage and creation, and implementing vector search*
- Automatic management of the model's context window
- Access multiple tools in parallel (up to 128 tools per Assistant) incl Code Interpreter
- Build your own tools using Function Calling

**coming soon*

Available Tools

Code Interpreter: Allows the Assistants API to write and run Python code in a sandboxed execution environment.

Knowledge Retrieval: Retrieval augments the Assistant with knowledge from outside its model. Once a file is uploaded and passed to the Assistant, OpenAI will automatically chunk your documents, index and store the embeddings.

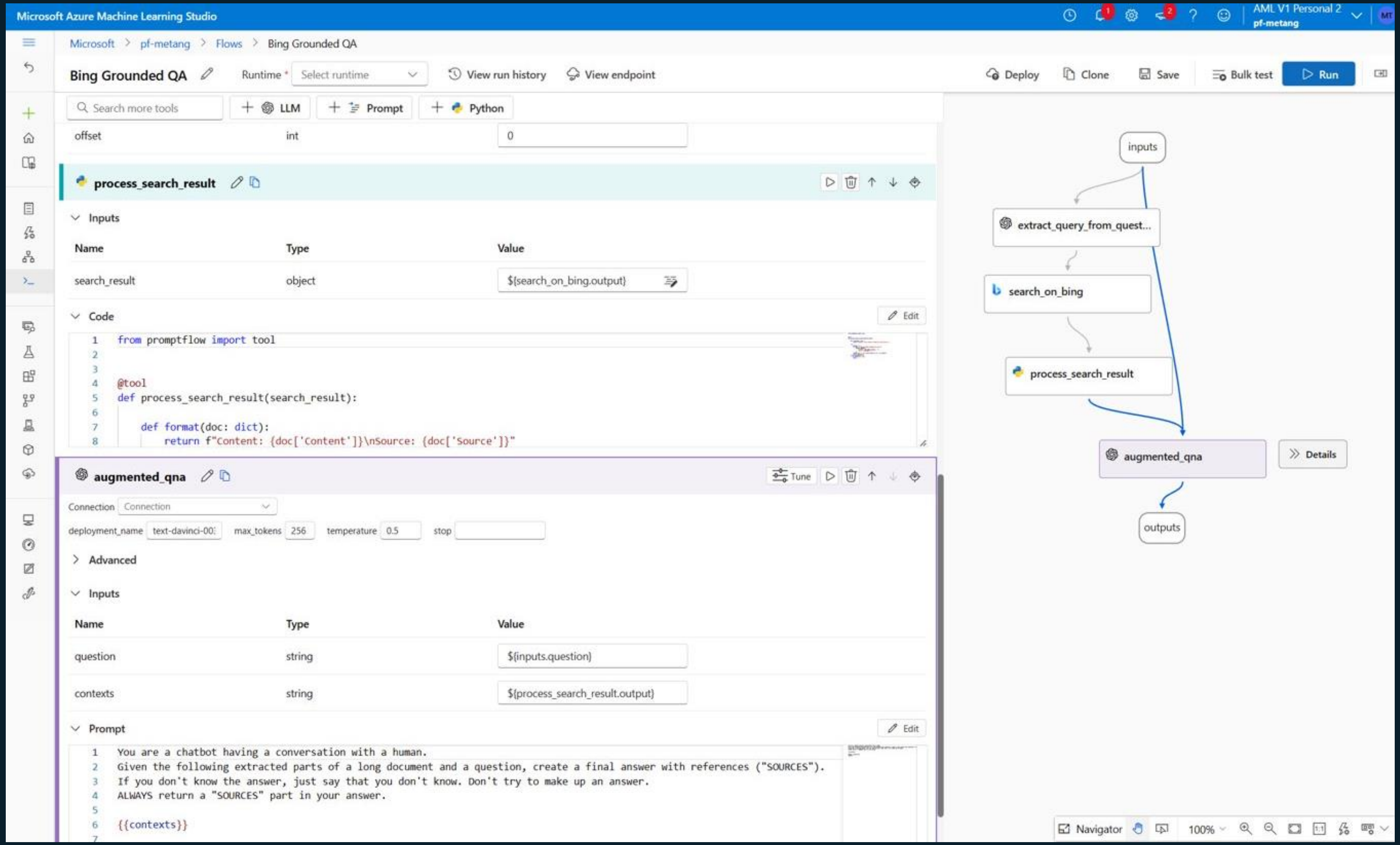
Function calling: Allows you to describe functions to the Assistants and have it intelligently return the functions that need to be called along with their arguments.

Prompt Flow

Azure Machine Learning prompt flow (1/7)

Capabilities Overview

- Develop workflows
  - Develop flows that connect to various language models, external data sources, tools, and custom code
- Test and evaluate
  - Test flows with large datasets in parallel
  - Evaluate the AI quality of the workflows with metrics like performance, groundedness, and accuracy
- Prompt tuning
  - Easily tune prompts with variants and versions
- Compare and deploy
  - Visually compare across experiments
  - One-click deploy to a managed endpoint for rapid integration

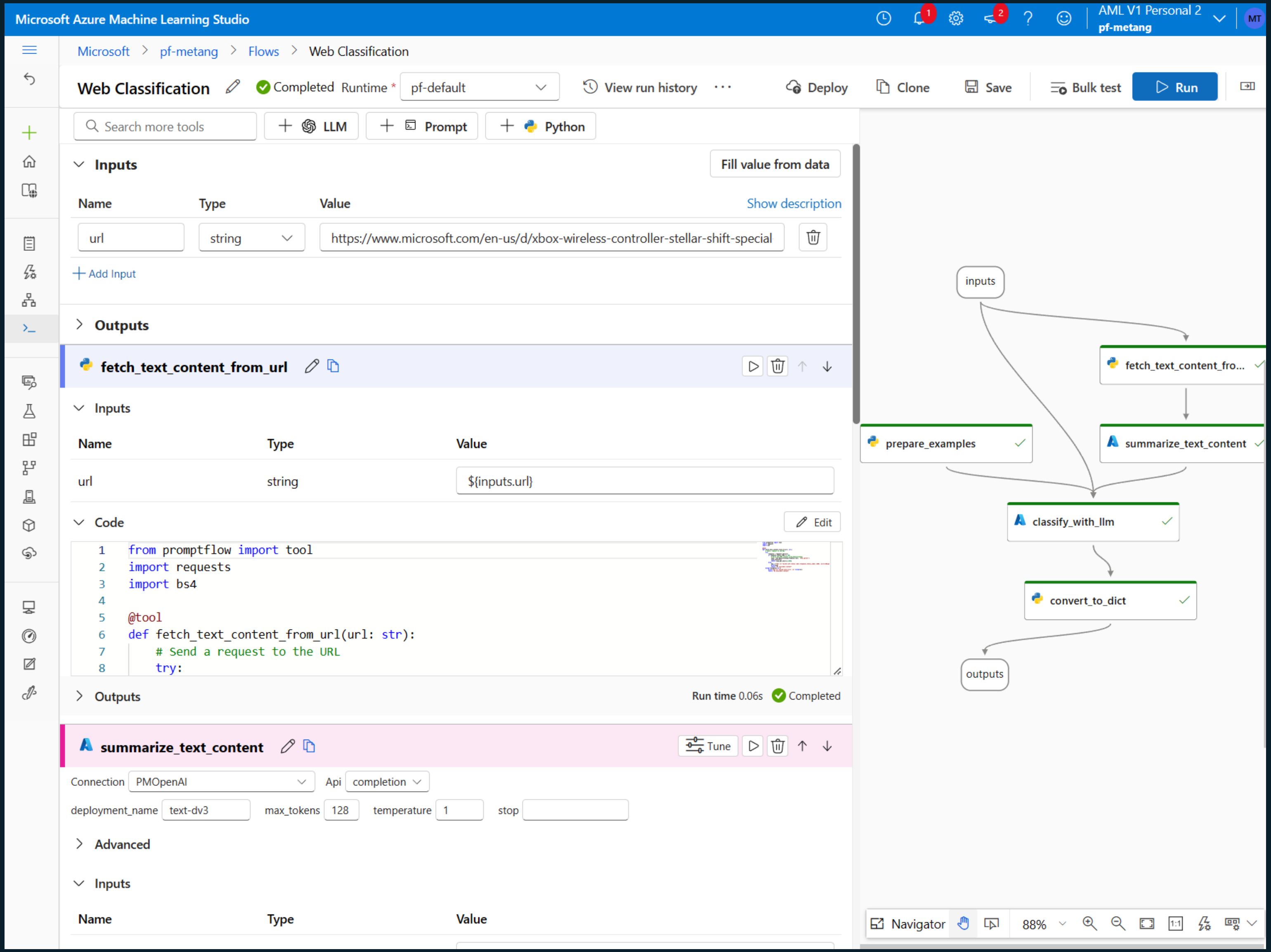


Azure Machine Learning prompt flow (2/7)

Prompt flow authoring

Develop your LLM flow from scratch

- Construct a flow using pre-built tools
- Support custom code
- Clone flows from samples
- Track run history

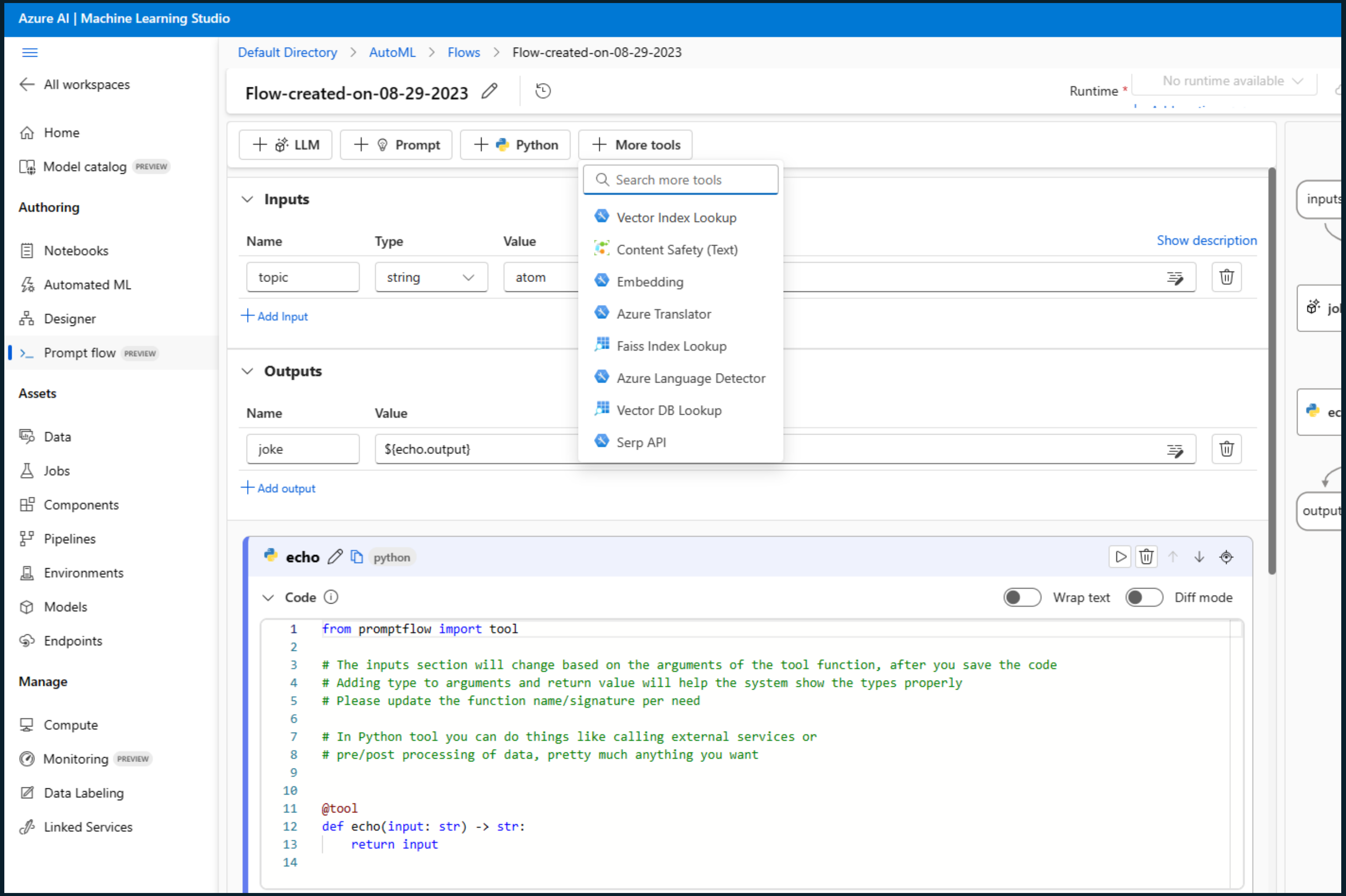


Azure Machine Learning prompt flow (3/7)

Connections

Manage APIs and external data sources

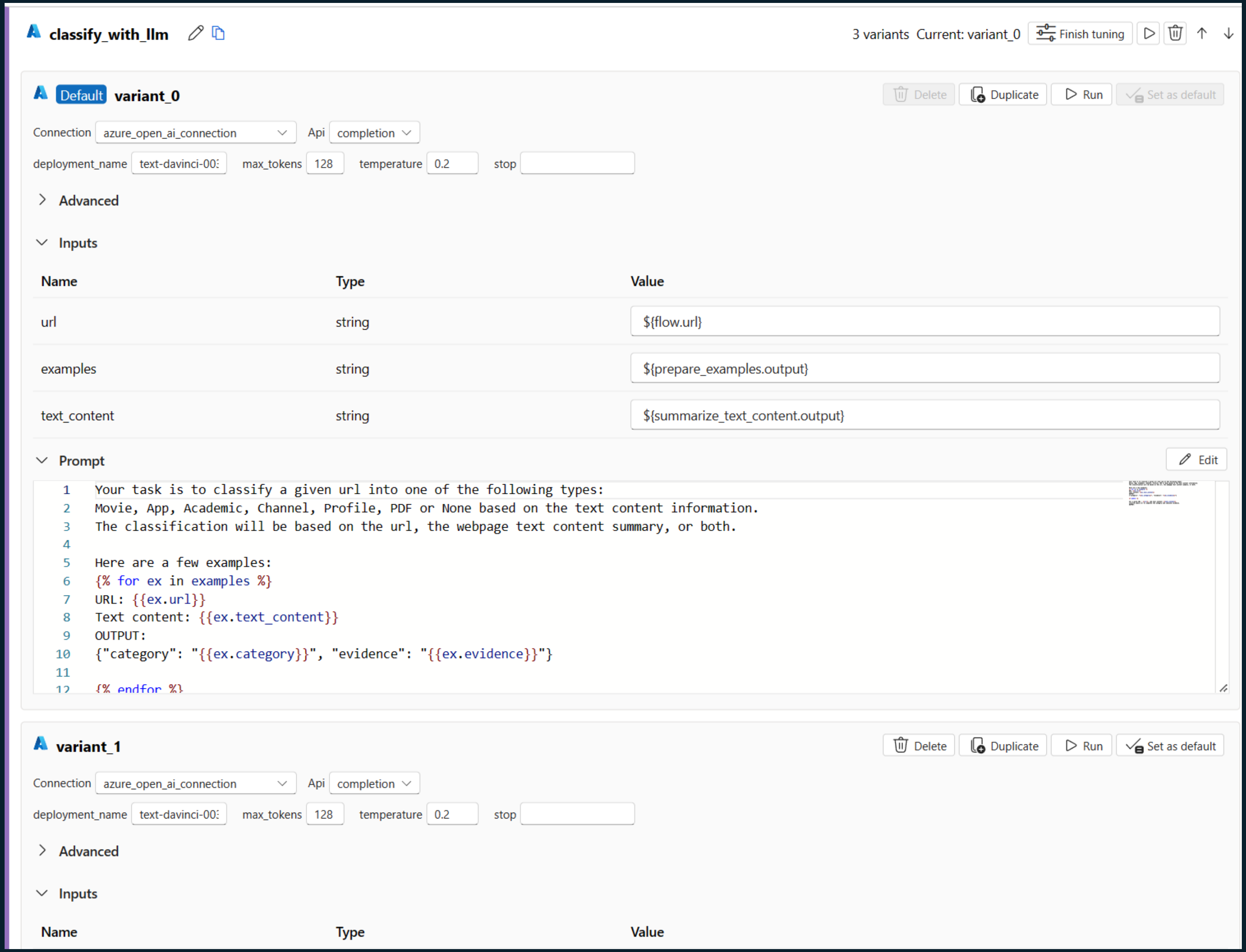
- Seamless integration with pre-built LLMs like Azure OpenAI Service
- Built-in safety system with Azure AI Content Safety
- Effectively manage credentials or secrets for APIs
- Create your own connections in Python tools



Azure Machine Learning prompt flow (4/7)

Variants

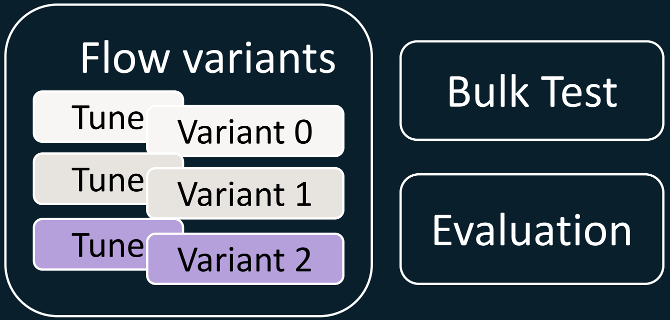
- Create dynamic prompts using external data and few shot samples
- Edit your complex prompts in full screen
- Quickly tune prompt and LLM configuration with variants

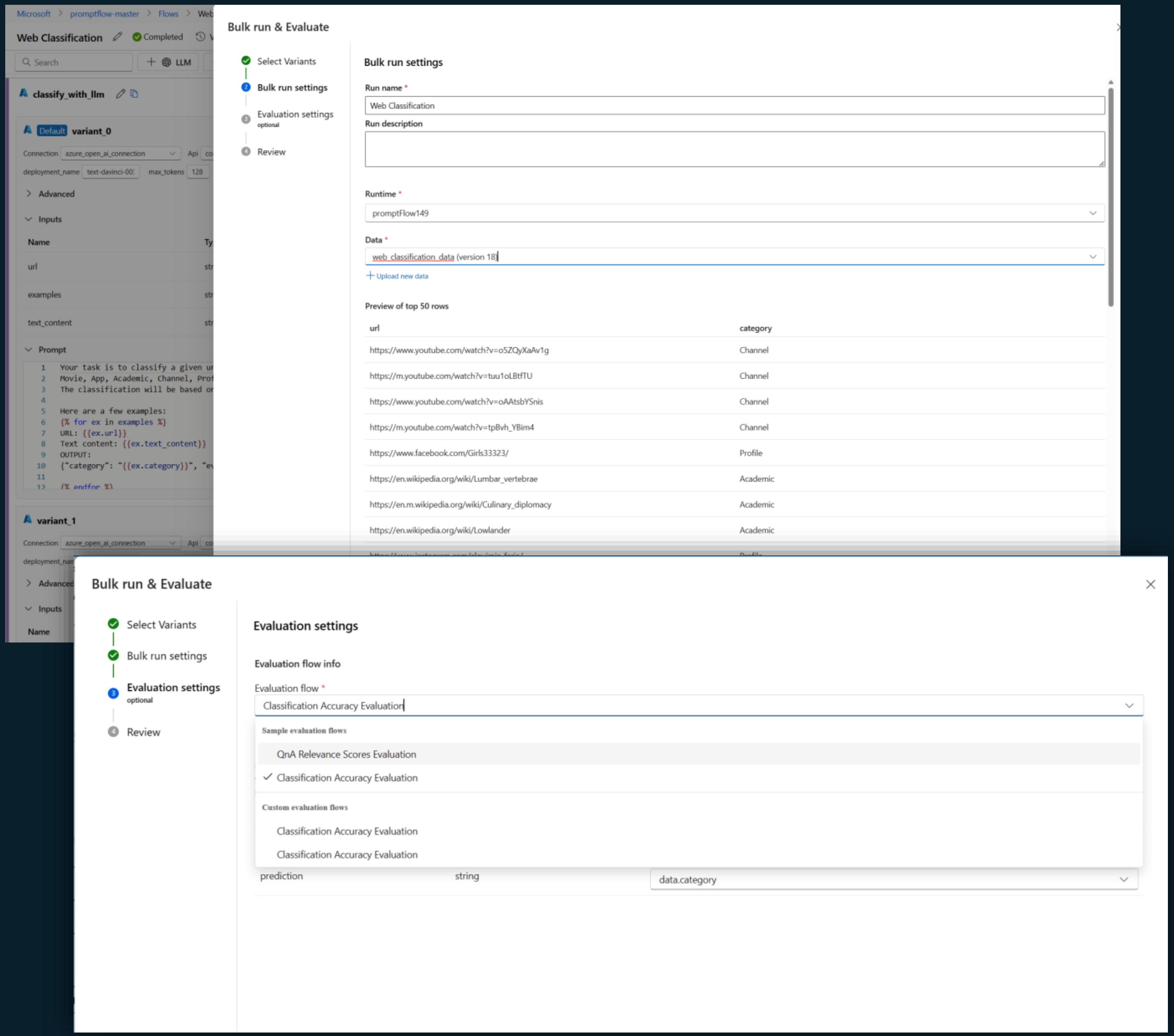


Azure Machine Learning prompt flow (5/7)

Evaluation

- Evaluate flow performance with your own data
- Use pre-built evaluation flows
- Build your own custom evaluation flows

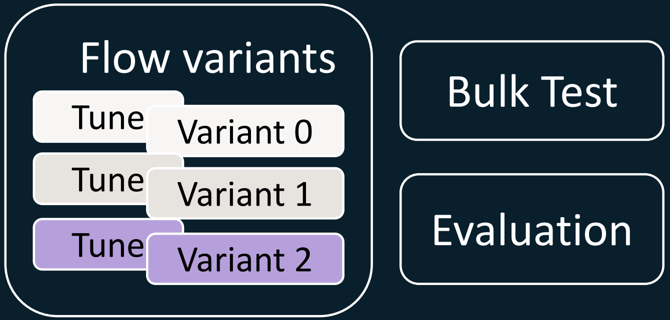


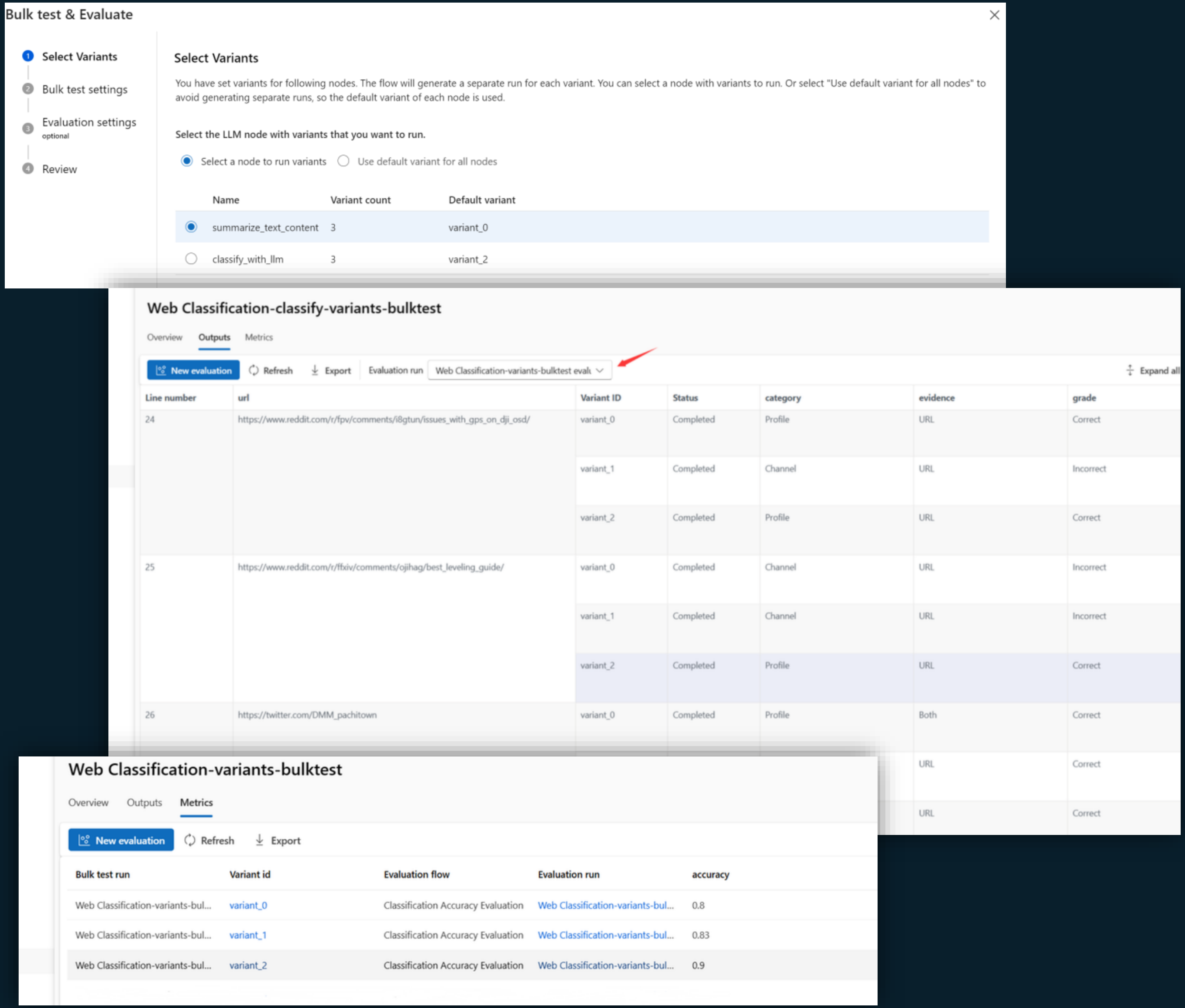


Azure Machine Learning prompt flow (6/7)

Evaluation

- Compare multiple variants or runs to pick best flow
- Add new evaluations to a finished run
- Ensure accuracy by scaling the size of data in evaluation

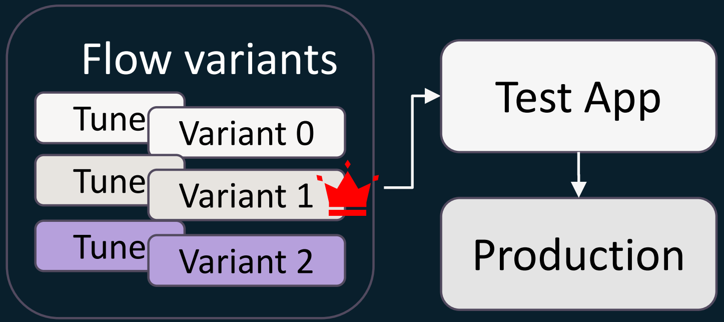


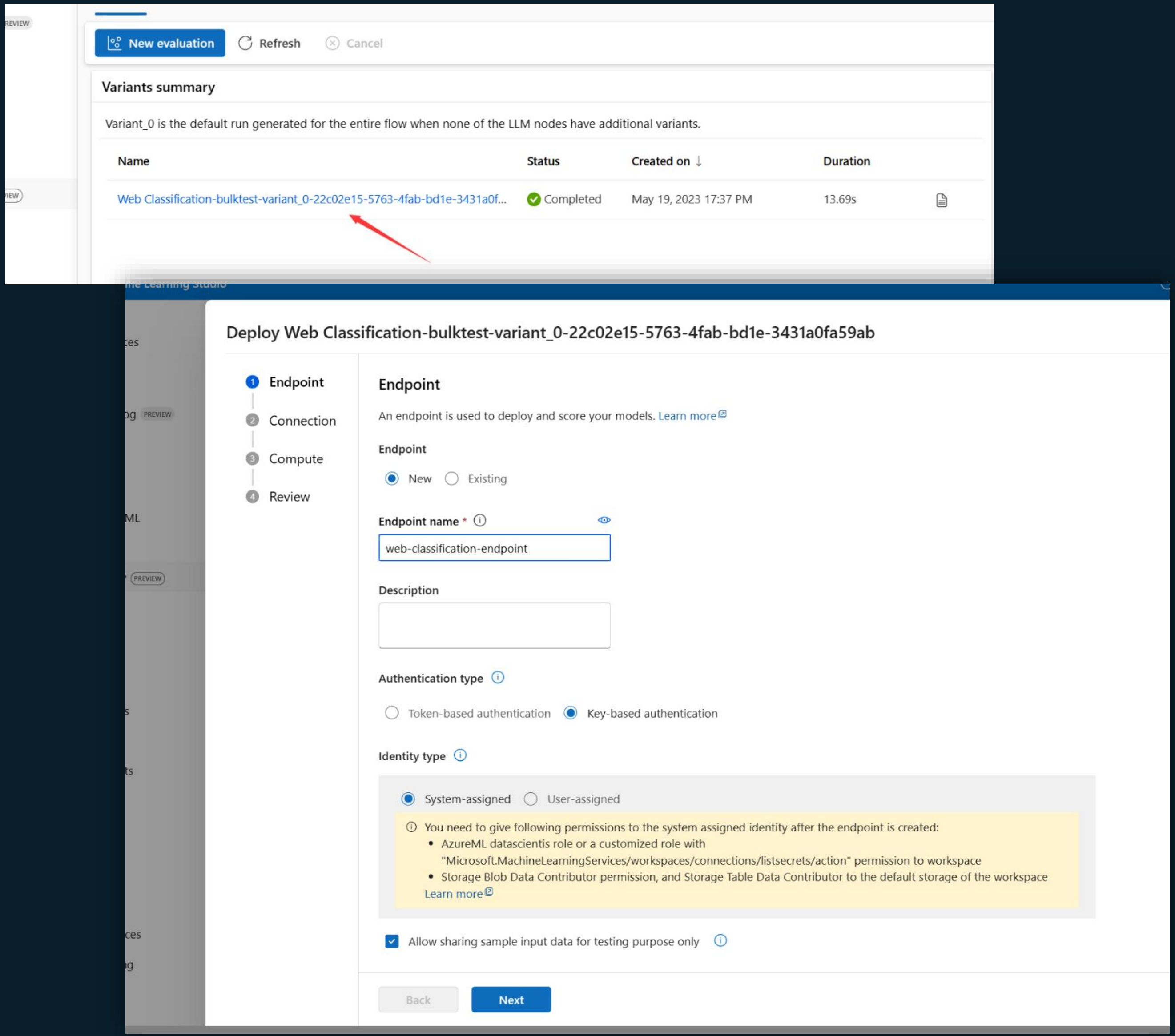


Azure Machine Learning prompt flow (7/7)

Deploy

- Seamless transition from development to production with AzureML's managed online endpoints





Let’ code…

Repos

Assistant Bot

Mangu/openai_assistant-bot (github.com)

Simple Chat Web Client

Mangu/simple-chat-webclient (github.com)