

# Building AI Agents / Copilots

Dani Diaz

Principal Technical Architect

Global Black Belt – Microsoft AI

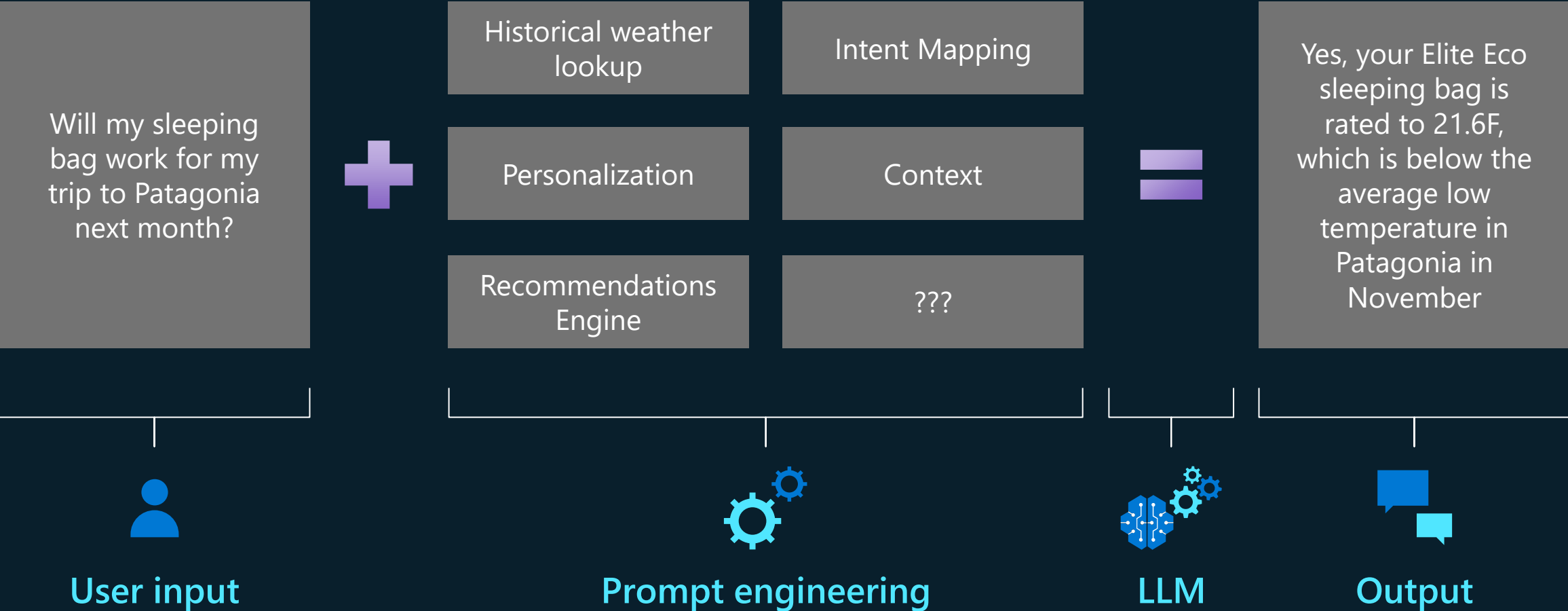
# LLMs are **like** Language Calculator

Prompt:

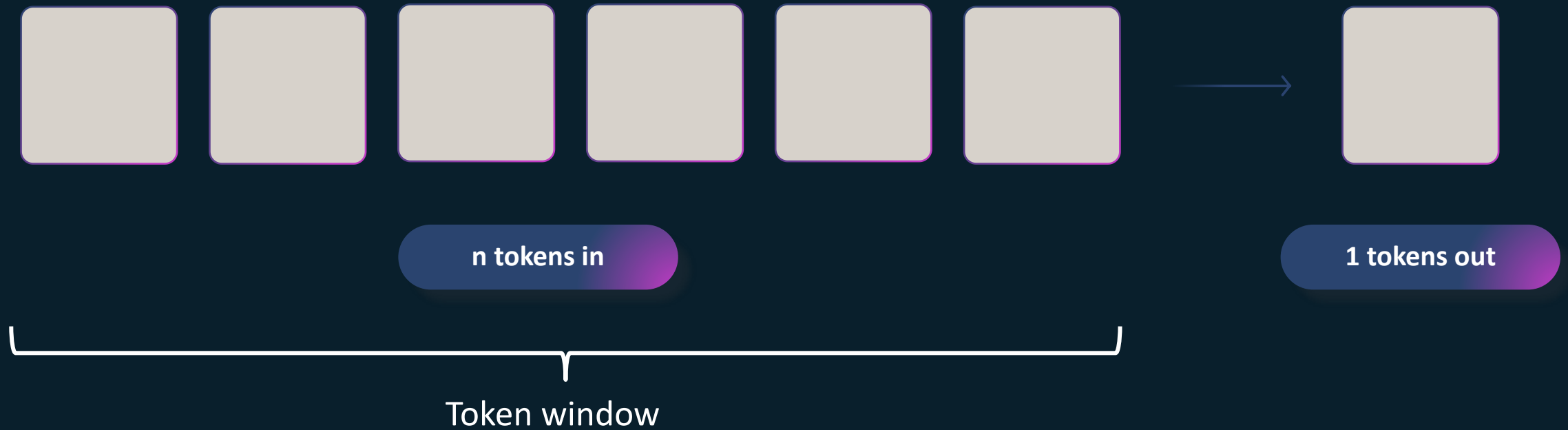
A realistic photo of a less futuristic language calculator on a desk in a classroom with a pencil and a spiral notebook



# Language Calculator



# 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](http://www.espn.com) Class: Sports

Page: [www.google.com](http://www.google.com) Class: Search

Classify the following page:

Page: [www.allrecipes.com](http://www.allrecipes.com) Class:

## Response

Cooking

## RAG

You know the following:  
[vectors]

Answer the following question:

How many US paid holidays are there?

## Response

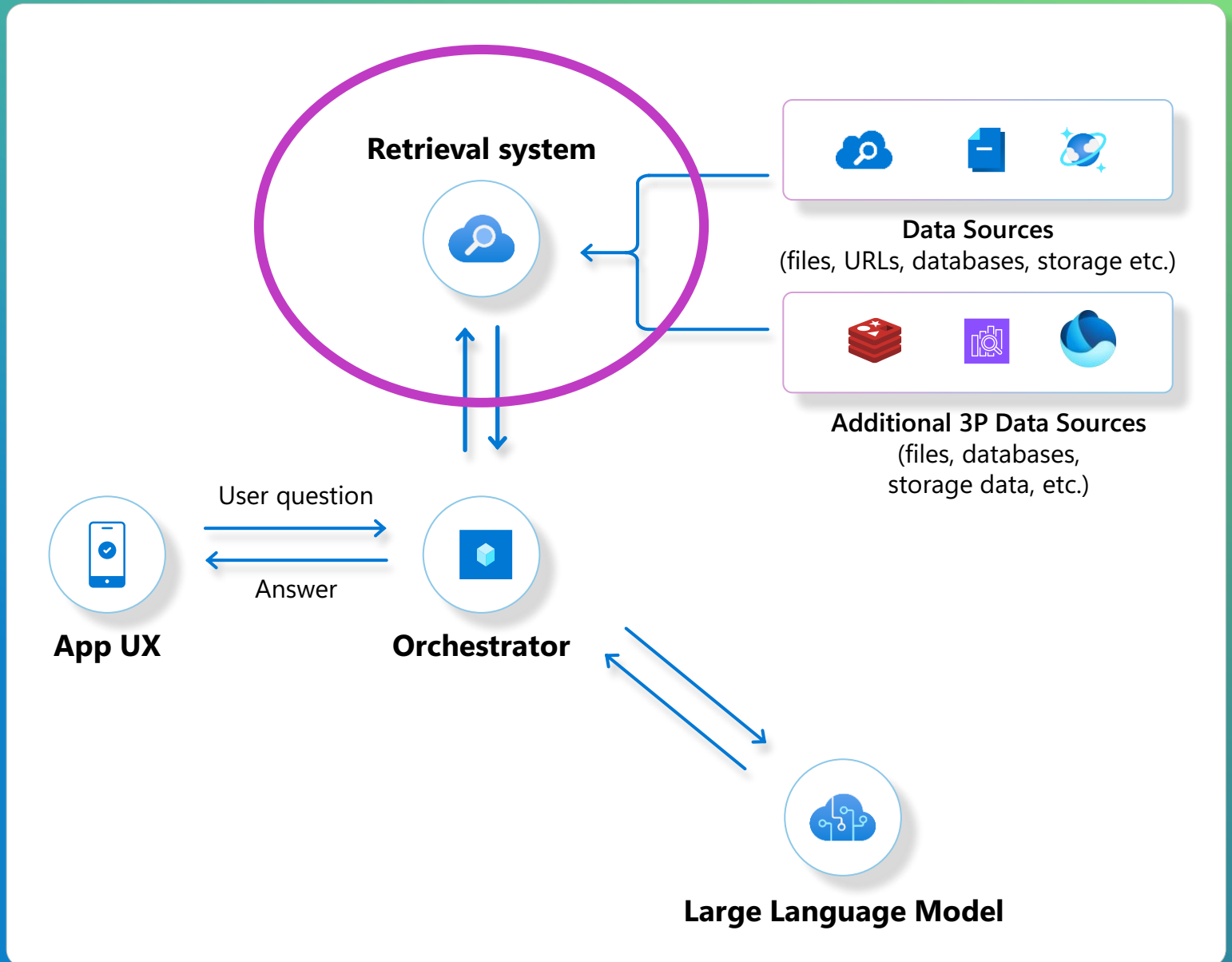
There are 10 paid holidays in the US.

# 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

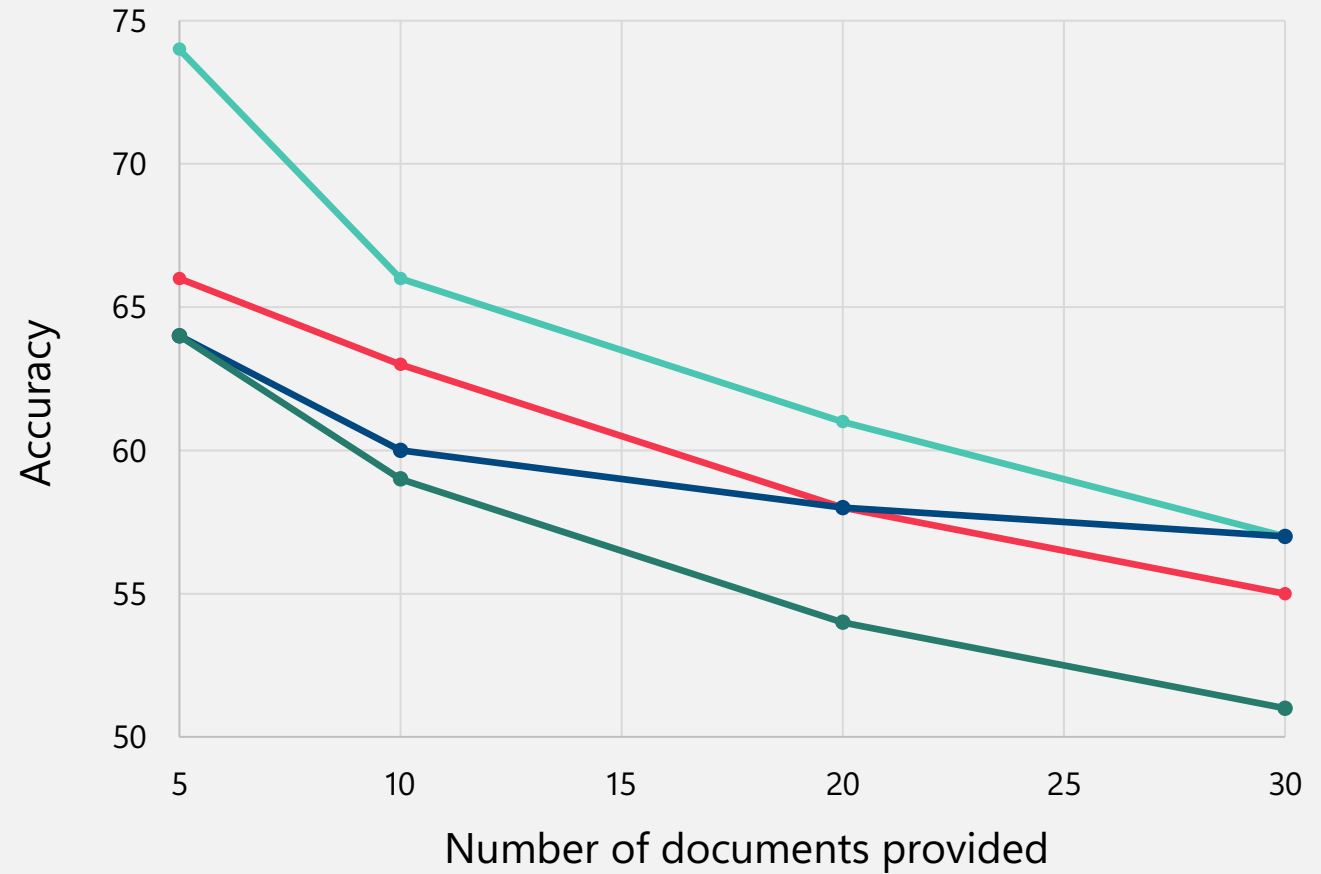


## **Retrieval augmentation**

Learn new facts

# Your retrieval strategy matters

More information  $\neq$   
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?”

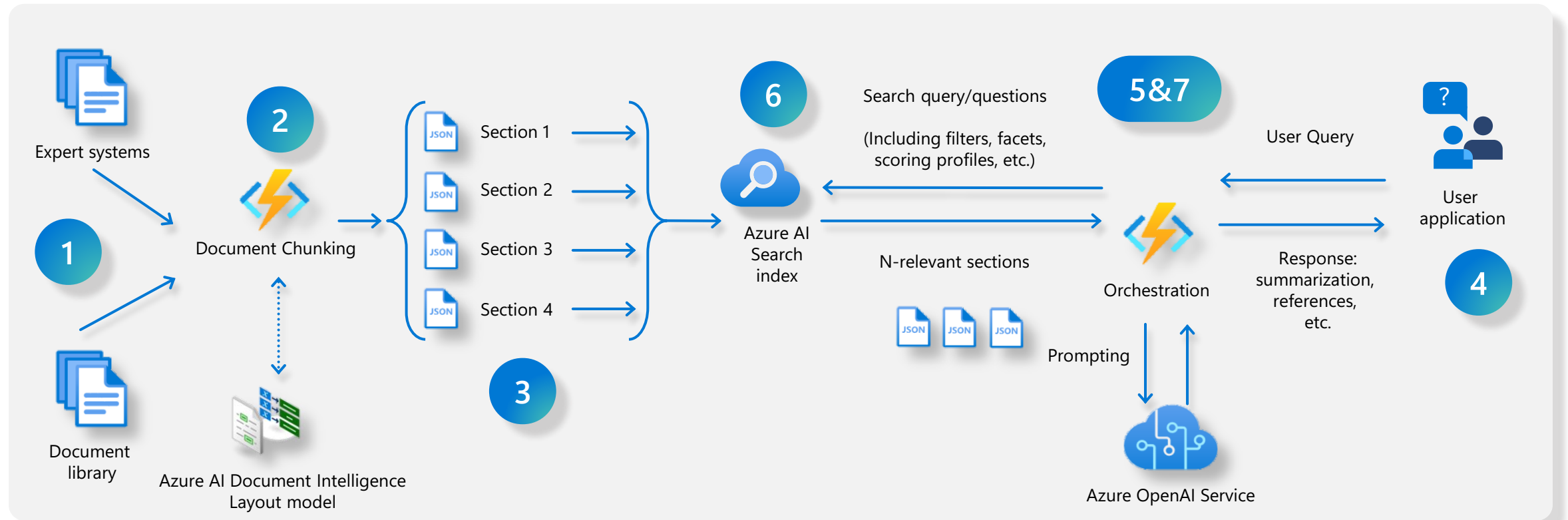
## Won't match:

“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

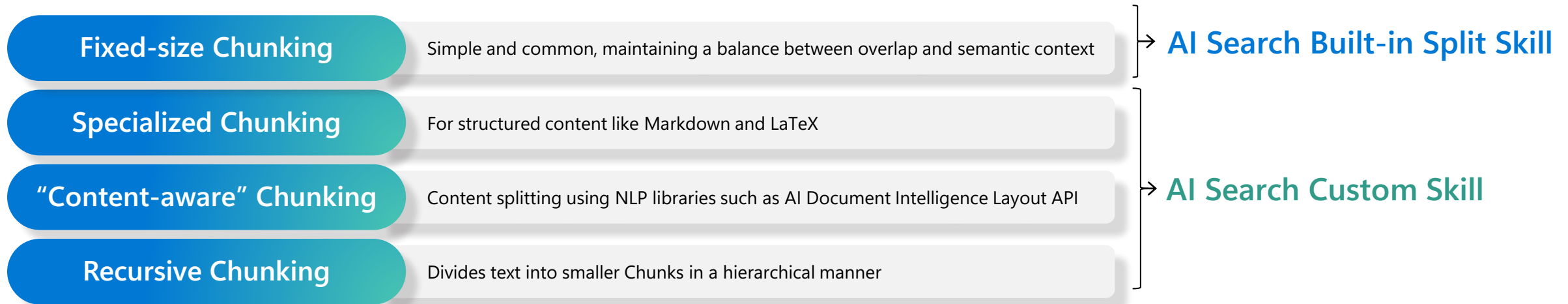
Retrieval Configuration	Single vector per document <i>[Recall@50]</i>	Chunked documents <i>[Recall@50]</i>
Queries whose answer is in long documents	28.2	<b>45.7</b>
Queries whose answer is deep into a document	28.7	<b>51.4</b>

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	<b>43.9</b>

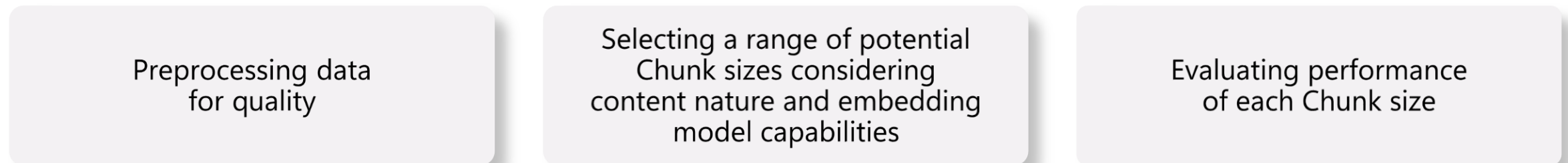
Retrieval Configuration	Recall@50
512 input tokens per vector	<b>42.4</b>
1024 input tokens per vector	37.5
4096 input tokens per vector	36.4
8191 input tokens per vector	34.9

# Chunking strategies

## Chunking methods



## Determining the best Chunk size



[!\[\]\(a03a7eb2f4046e1d3c76772003e549ea\_img.jpg\) 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

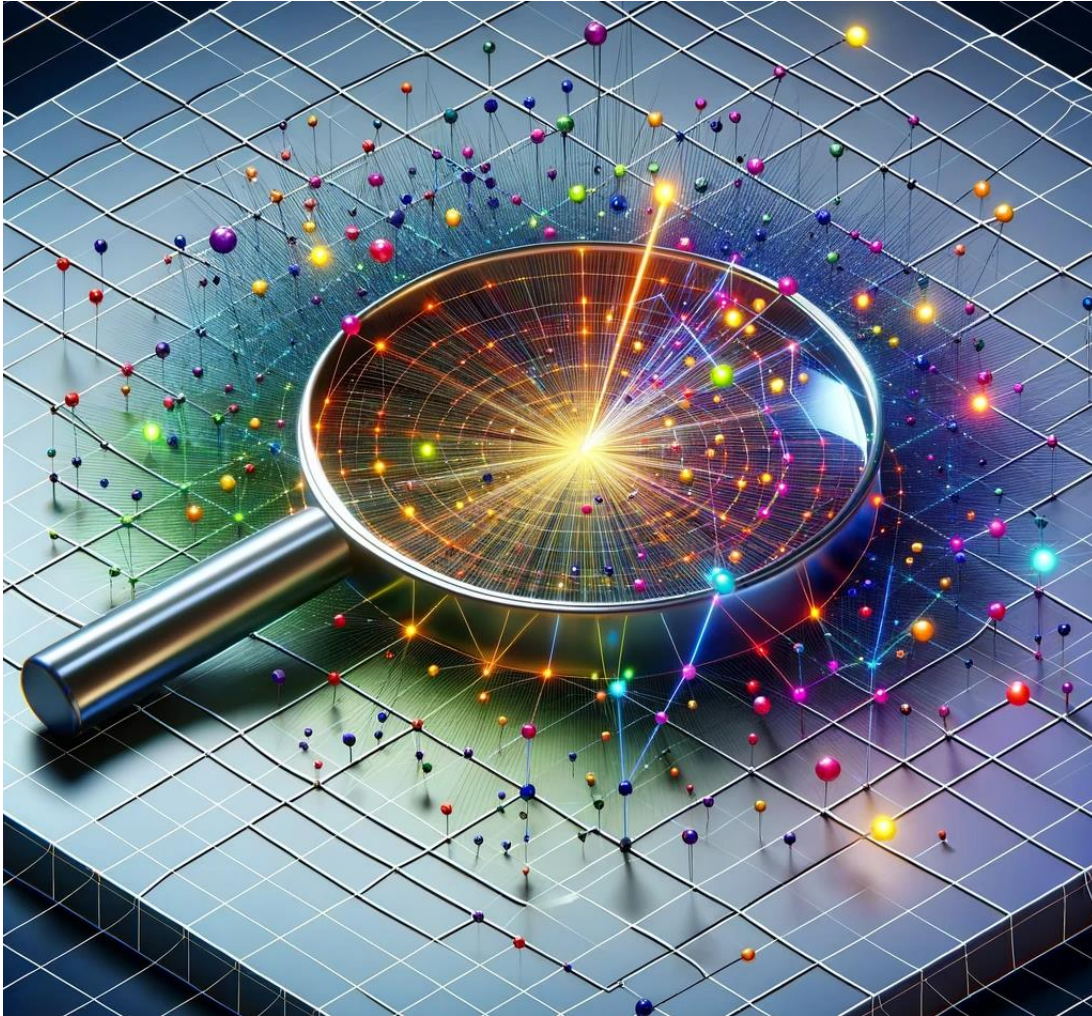
### **What is Semantic ranking?**

- 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  $\rightarrow$  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

Apps

Microsoft Copilot Extensibility

Copilots

Microsoft Copilots  
+ your copilots

AI orchestration

Instructions (Meta Prompt)

Function Calling

Code Interpreter

Knowledge Retrieval

Assistant  
Tools

Azure OpenAI  
Assistants API

Built-in  
safety  
system  
and  
responsible  
AI  
tools

Foundation models

AI infrastructure

# 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**

## Assistant

Personal Finance bot

## Instructions

You are a personal finance advisor chatbot. Use your knowledge base to best respond to customer queries

## Model

gpt-3.5-turbo or gpt-4 models

## Tools (optional)

File upload (bank statements, investment statements, loan documents, etc.)  
Code Interpreter  
Retrieval  
Functions

## Thread

Retirement Planning

### User's message

How much should I contribute to my retirement plan?

### Assistant's message

You should contribute \$478 per year

## Run 1

**Assistant** Personal finance bot  
**Thread** Retirement planning  
**Steps**

1

Use code interpreter with files retrieved

2

Create message

## Run 2

# 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.

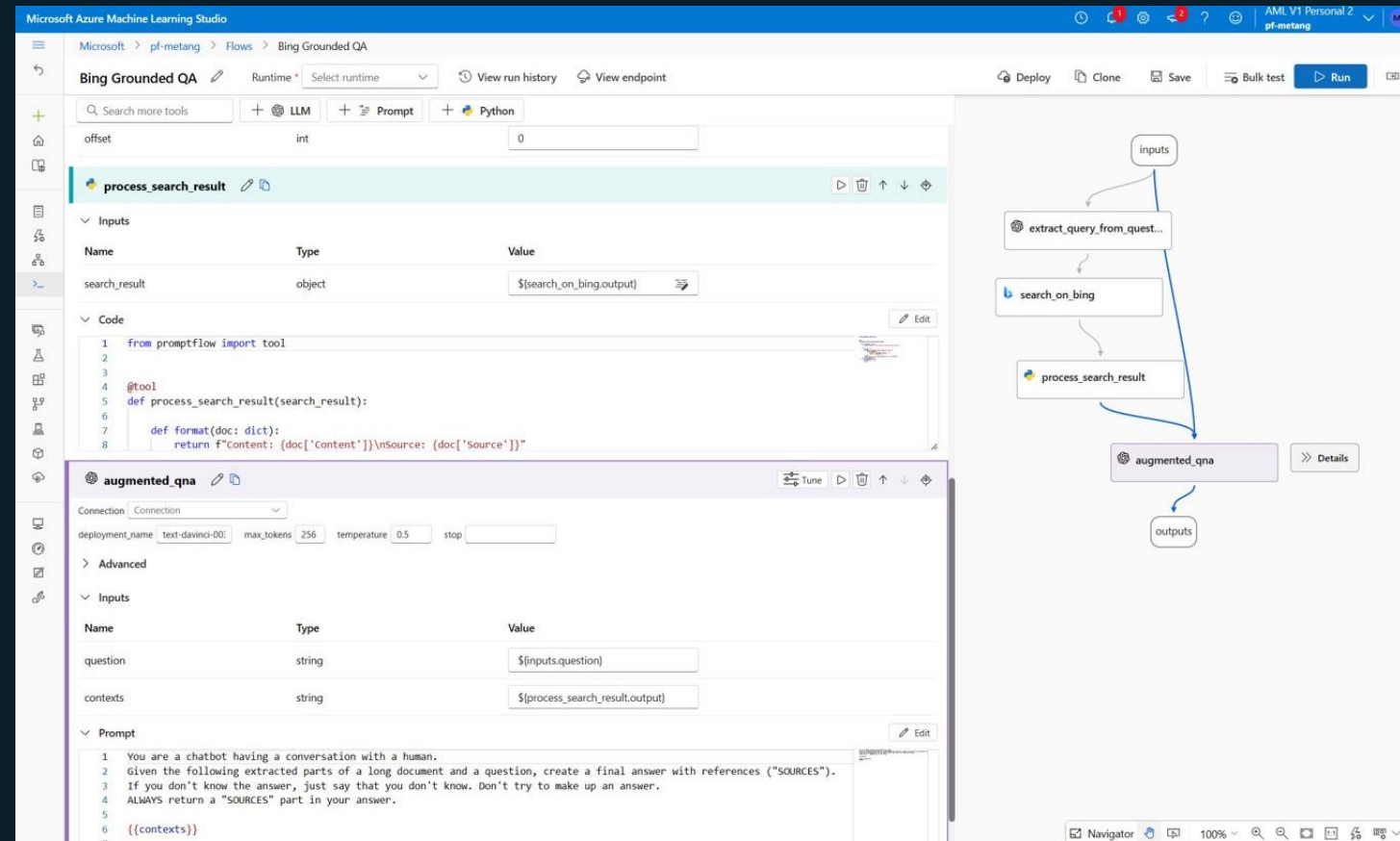
**Function calling:** Allows you to describe functions to the Assistants and have it intelligently return the functions that need to be called along with their arguments.

# 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

Microsoft Azure Machine Learning Studio

Microsoft > pf-metang > Flows > Web Classification

Web Classification Completed Runtime \* pf-default View run history Deploy Clone Save Bulk test Run

Search more tools + LLM + Prompt + Python

**Inputs** Fill value from data Show description

Name	Type	Value
url	string	https://www.microsoft.com/en-us/d/xbox-wireless-controller-stellar-shift-special

+ Add Input

**Outputs**

**fetch\_text\_content\_from\_url** Run Copy Up Down

**Inputs**

Name	Type	Value
url	string	\${inputs.url}

**Code** Edit

```
1 from promptflow import tool
2 import requests
3 import bs4
4
5 @tool
6 def fetch_text_content_from_url(url: str):
7     # Send a request to the URL
8     try:
```

**Outputs** Run time 0.06s Completed

**summarize\_text\_content** Tune Run Copy Up Down

Connection PMOpenAI Api completion

deployment\_name text-dv3 max\_tokens 128 temperature 1 stop

**Advanced**

**Inputs**

Name	Type	Value
------	------	-------

Flow diagram: inputs → prepare\_examples → fetch\_text\_content\_from\_url → summarize\_text\_content → classify\_with\_llm → convert\_to\_dict → outputs

Navigator 88% Search Zoom Reset Fullscreen

# 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

The screenshot displays the Azure AI Machine Learning Studio interface. The top navigation bar shows the path: Default Directory > AutoML > Flows > Flow-created-on-08-29-2023. The main workspace is titled "Flow-created-on-08-29-2023" and includes a "Runtime" dropdown set to "No runtime available".

On the left, a sidebar contains navigation links: All workspaces, Home, Model catalog (PREVIEW), Authoring (Notebooks, Automated ML, Designer, Prompt flow (PREVIEW)), Assets (Data, Jobs, Components, Pipelines, Environments, Models, Endpoints), and Manage (Compute, Monitoring (PREVIEW), Data Labeling, Linked Services).

The central area shows the "Inputs" and "Outputs" sections. The "Inputs" table has columns for Name, Type, and Value, with one input named "topic" of type "string" and value "atom". The "Outputs" table has columns for Name and Value, with one output named "joke" and value "\${echo.output}".

A dropdown menu is open, showing a search bar "Search more tools" and a list of available tools: Vector Index Lookup, Content Safety (Text), Embedding, Azure Translator, Faiss Index Lookup, Azure Language Detector, Vector DB Lookup, and Serp API. A "Show description" link is visible next to the Content Safety (Text) tool.

At the bottom, a code editor window titled "echo" shows a Python script:

```
1 from promptflow import tool
2
3 # The inputs section will change based on the arguments of the tool function, after you save the code
4 # Adding type to arguments and return value will help the system show the types properly
5 # Please update the function name/signature per need
6
7 # In Python tool you can do things like calling external services or
8 # pre/post processing of data, pretty much anything you want
9
10
11 @tool
12 def echo(input: str) -> str:
13     return input
14
```

The code editor includes a "Code" tab, a "Wrap text" toggle, and a "Diff mode" toggle.

# 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

The screenshot displays the Azure Machine Learning prompt flow interface. At the top, it shows the flow name 'classify\_with\_llm' and the current variant 'variant\_0'. The interface is divided into two main sections: 'variant\_0' and 'variant\_1'.

**variant\_0 configuration:**

- Connection:** azure\_open\_ai\_connection
- Api:** completion
- deployment\_name:** text-davinci-001
- max\_tokens:** 128
- temperature:** 0.2
- stop:** (empty field)

**Inputs table:**

Name	Type	Value
url	string	\$(flow.url)
examples	string	\$(prepare_examples.output)
text_content	string	\$(summarize_text_content.output)

**Prompt:**

```
1 Your task is to classify a given url into one of the following types:
2 Movie, App, Academic, Channel, Profile, PDF or None based on the text content information.
3 The classification will be based on the url, the webpage text content summary, or both.
4
5 Here are a few examples:
6 {% for ex in examples %}
7 URL: {{ex.url}}
8 Text content: {{ex.text_content}}
9 OUTPUT:
10 {"category": "{{ex.category}}", "evidence": "{{ex.evidence}}"}
11
12 {% endfor %}
```

**variant\_1 configuration:**

- Connection:** azure\_open\_ai\_connection
- Api:** completion
- deployment\_name:** text-davinci-001
- max\_tokens:** 128
- temperature:** 0.2
- stop:** (empty field)

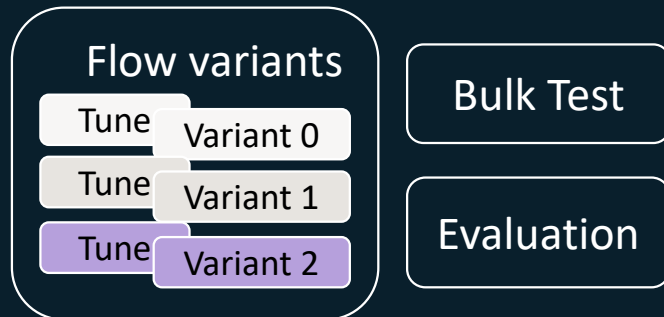
**Inputs table:**

Name	Type	Value
------	------	-------

# 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



Microsoft > promptflow-master > Flows > Web

Web Classification Completed

Search + LLM

**classify\_with\_llm**

**variant\_0**

Connection: azure\_open\_ai\_connection | Deployment name: text-davinci-002 | Max tokens: 128

Advanced

Inputs

Name	Type
url	string
examples	string
text_content	string

Prompt

```
1 Your task is to classify a given url
2 Movie, App, Academic, Channel, Profile
3 The classification will be based on
4
5 Here are a few examples:
6 [% for ex in examples %]
7 URL: [% (ex.url) %]
8 Text content: [% (ex.text_content) %]
9 OUTPUT:
10 {"category": "[% (ex.category) %]"}, "examples": [% (ex.examples) %]
11
12 [% (ex.url) %]
```

**variant\_1**

Connection: azure\_open\_ai\_connection | Deployment name: text-davinci-002 | Max tokens: 128

Advanced

Inputs

Name

**Bulk run & Evaluate**

Select Variants

Bulk run settings

Evaluation settings optional

Review

**Bulk run settings**

Run name \*  
Web Classification

Run description

Runtime \*  
promptflow149

Data \*  
web\_classification\_data (version 10)  
[+ Upload new data](#)

Preview of top 50 rows

url	category
https://www.youtube.com/watch?v=o5ZQyKaVv1g	Channel
https://my.youtube.com/watch?v=tuu1oL8tFTU	Channel
https://www.youtube.com/watch?v=oAAtsbY5nIs	Channel
https://my.youtube.com/watch?v=tpBvh_YBim4	Channel
https://www.facebook.com/Girls33323/	Profile
https://en.wikipedia.org/wiki/Lumbar_vertebrae	Academic
https://en.m.wikipedia.org/wiki/Culinary_diplomacy	Academic
https://en.wikipedia.org/wiki/Lowlander	Academic

**Bulk run & Evaluate**

Select Variants

Bulk run settings

Evaluation settings optional

Review

**Evaluation settings**

Evaluation flow info

Evaluation flow \*  
Classification Accuracy Evaluation

Sample evaluation flows

- QnA Relevance Scores Evaluation
- ✓ Classification Accuracy Evaluation

Custom evaluation flows

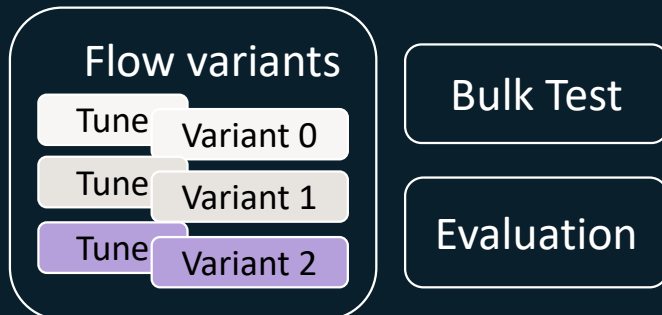
- Classification Accuracy Evaluation
- Classification Accuracy Evaluation

prediction string data.category

# 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



The screenshot displays the 'Bulk test & Evaluate' interface in Azure Machine Learning. The top section, 'Select Variants', shows a list of LLM nodes with their variant counts and default variants. The 'summarize\_text\_content' node is selected with 3 variants (variant\_0, variant\_1, variant\_2). The 'classify\_with\_llm' node is also listed with 3 variants (variant\_0, variant\_1, variant\_2).

The bottom section, 'Web Classification-classify-variants-bulktest', shows a table of evaluation results. A red arrow points to the 'New evaluation' button. The table has columns: Line number, url, Variant ID, Status, category, evidence, and grade.

Line number	url	Variant ID	Status	category	evidence	grade
24	https://www.reddit.com/r/fpv/comments/8gtun/issues_with_gps_on_dji_osd/	variant_0	Completed	Profile	URL	Correct
		variant_1	Completed	Channel	URL	Incorrect
		variant_2	Completed	Profile	URL	Correct
25	https://www.reddit.com/r/fpv/comments/ojhag/best_leveling_guide/	variant_0	Completed	Channel	URL	Incorrect
		variant_1	Completed	Channel	URL	Incorrect
		variant_2	Completed	Profile	URL	Correct
26	https://twitter.com/DMM_pachitown	variant_0	Completed	Profile	Both	Correct

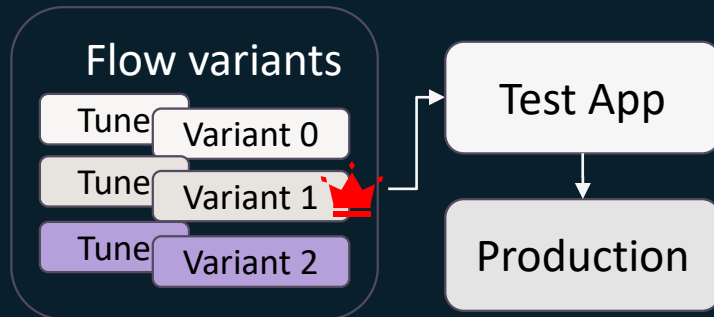
The bottom section, 'Web Classification-variants-bulktest', shows a table of evaluation results. A red arrow points to the 'New evaluation' button. The table has columns: Bulk test run, Variant id, Evaluation flow, Evaluation run, and accuracy.

Bulk test run	Variant id	Evaluation flow	Evaluation run	accuracy
Web Classification-variants-bul...	variant_0	Classification Accuracy Evaluation	Web Classification-variants-bul...	0.8
Web Classification-variants-bul...	variant_1	Classification Accuracy Evaluation	Web Classification-variants-bul...	0.83
Web Classification-variants-bul...	variant_2	Classification Accuracy Evaluation	Web Classification-variants-bul...	0.9

# Azure Machine Learning prompt flow (7/7)

## Deploy

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



The screenshot shows the 'New evaluation' table in the Azure ML interface. It includes buttons for 'New evaluation', 'Refresh', and 'Cancel'. Below the buttons is a 'Variants summary' section stating that 'Variant\_0' is the default run. A table lists the evaluation runs with columns for Name, Status, Created on, and Duration. A red arrow points to the first row, which is 'Web Classification-bulktest-variant\_0-22c02e15-5763-4fab-bd1e-3431a0f...' with a status of 'Completed'.

Name	Status	Created on ↓	Duration
Web Classification-bulktest-variant_0-22c02e15-5763-4fab-bd1e-3431a0f...	Completed	May 19, 2023 17:37 PM	13.69s

The screenshot shows the 'Deploy Web Classification-bulktest-variant\_0-22c02e15-5763-4fab-bd1e-3431a0fa59ab' form. It has a sidebar with steps: 1. Endpoint, 2. Connection, 3. Compute, and 4. Review. The 'Endpoint' step is active. The form includes fields for 'Endpoint name' (set to 'web-classification-endpoint') and 'Description'. Under 'Authentication type', 'Key-based authentication' is selected. Under 'Identity type', 'System-assigned' is selected. A yellow box provides instructions on the permissions required for the system-assigned identity. At the bottom, there is a checkbox for 'Allow sharing sample input data for testing purpose only' and 'Back'/'Next' buttons.

Let' code...

# Repos

Assistant Bot

[Mangu/openai\\_assistant-bot \(github.com\)](#)

Simple Chat Web Client

[Mangu/simple-chat-webclient \(github.com\)](#)