Microsoft

Microsoft Ignite

Microsoft

LAB515

Build an Agentic App with GraphRAG, Agent Framework, and the VS Code Extension for PostgreSQL

Speakers:
Varun Dhawan
Jonathon Frost
Jared Meade
Gauri Kasar
Soubhagya Dash

Meet the speakers and proctors

Varun Dhawan

Principal Product Manager, Microsoft

Jonathon Frost

Principal Technical Program Manager, Microsoft

Jared Meade

Senior Program Manager, Microsoft

Gauri Kasar

Product Manager, Microsoft

Soubhagya Dash

Principal PM Manager, Microsoft

Agenda

- Lab Overview
- PostgreSQL AI Core Concepts
- Lab “Follow Me”
  - Part 0 - Login to Azure
  - Part 1 - Setup your PostgreSQL Database
  - Part 2 - Use AI-driven features in PostgreSQL
  - Part 3 - Build the Agentic App

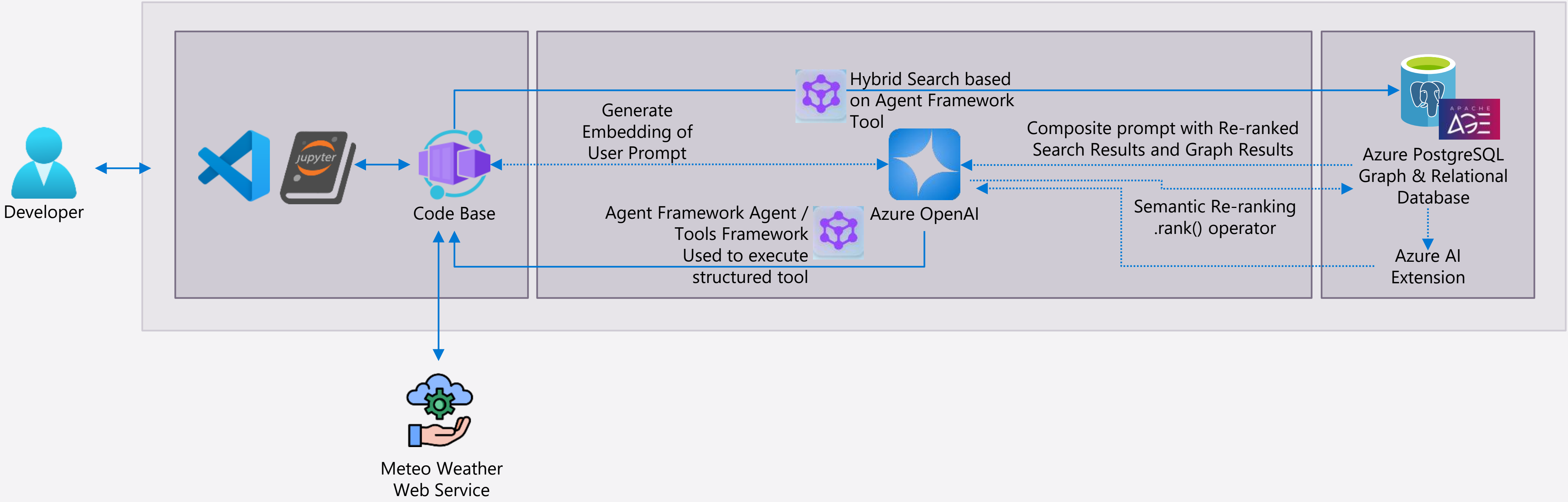
Lab Overview

What you will learn:

- How to use the VS Code PostgreSQL Extension
- Understand how to use Vector and Vector Indexes with PostgreSQL
- Learn about Agentic App architectures and coding patterns
- Hands-on building an Agentic App with PostgreSQL

Agentic App Architecture

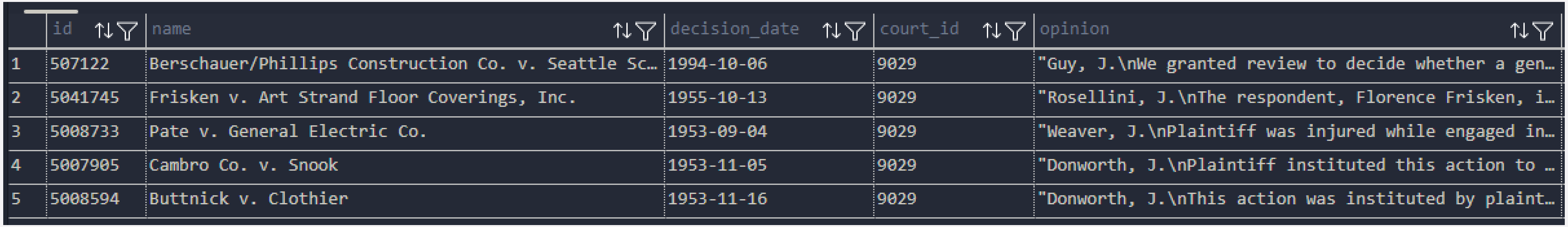
The App we are going to build today.



Developer
Code Base
Generate Embedding of User Prompt
Hybrid Search based on Agent Framework Tool
Composite prompt with Re-ranked Search Results and Graph Results
Azure PostgreSQL Graph & Relational Database
Agent Framework Agent / Tools Framework Used to execute structured tool
Azure OpenAI
Semantic Re-ranking .rank() operator
Azure AI Extension
Meteo Weather Web Service

Dataset for the Lab

- Caselaw Dataset for Washington State
- Subset of 337 unique legal cases
- Columns include: id, name, opinion, etc.
- Located at C:\Lab\Dataset\cases.csv



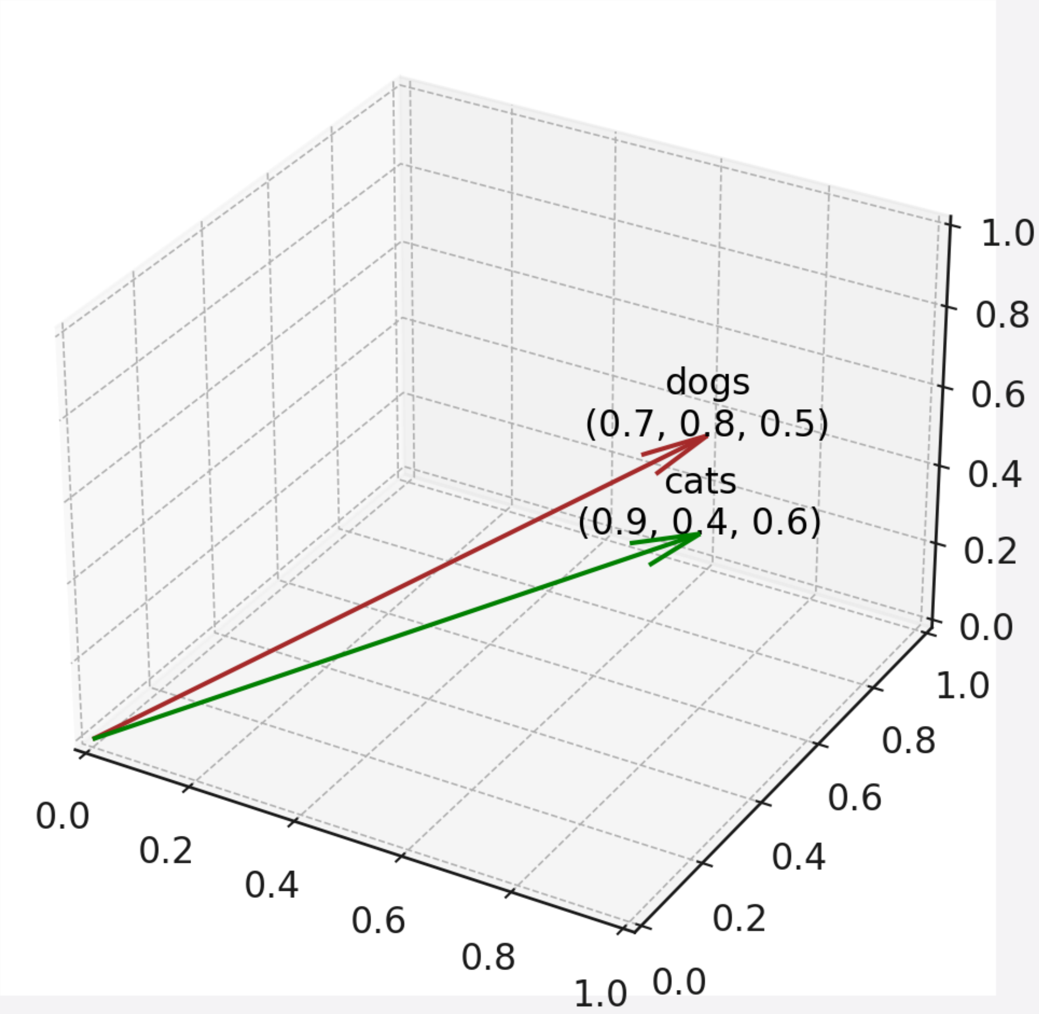
	id	name	decision_date	court_id	opinion
1	507122	Berschauer/Phillips Construction Co. v. Seattle Sc…	1994-10-06	9029	"Guy, J.\nWe granted review to decide whether a gen…
2	5041745	Frisken v. Art Strand Floor Coverings, Inc.	1955-10-13	9029	"Rosellini, J.\nThe respondent, Florence Frisken, i…
3	5008733	Pate v. General Electric Co.	1953-09-04	9029	"Weaver, J.\nPlaintiff was injured while engaged in…
4	5007905	Cambro Co. v. Snook	1953-11-05	9029	"Donworth, J.\nPlaintiff instituted this action to …
5	5008594	Buttnick v. Clothier	1953-11-16	9029	"Donworth, J.\nThis action was instituted by plaint…

PostgreSQL AI Core Concepts

- Vectors / Embeddings
- Vector Indexes
- Semantic Search

What is a Vector?

- Lists of numbers that represent items in a high-dimensional space.
- For example, a vector representing the string “**dogs**" might be [0.7, 0.8, 0.5].
- Each number in the vector is a dimension of the space.



How to generate a vector?

Use a model to generate vectors for items:

Input	→	Model	→	Vector
"**dog**"		text-embedding-3-small		[0.017198, -0.007493, -0.057982, ..]
"**cat**"		text-embedding-3-small		[0.004059, 0.06719, -0.093874, ...]

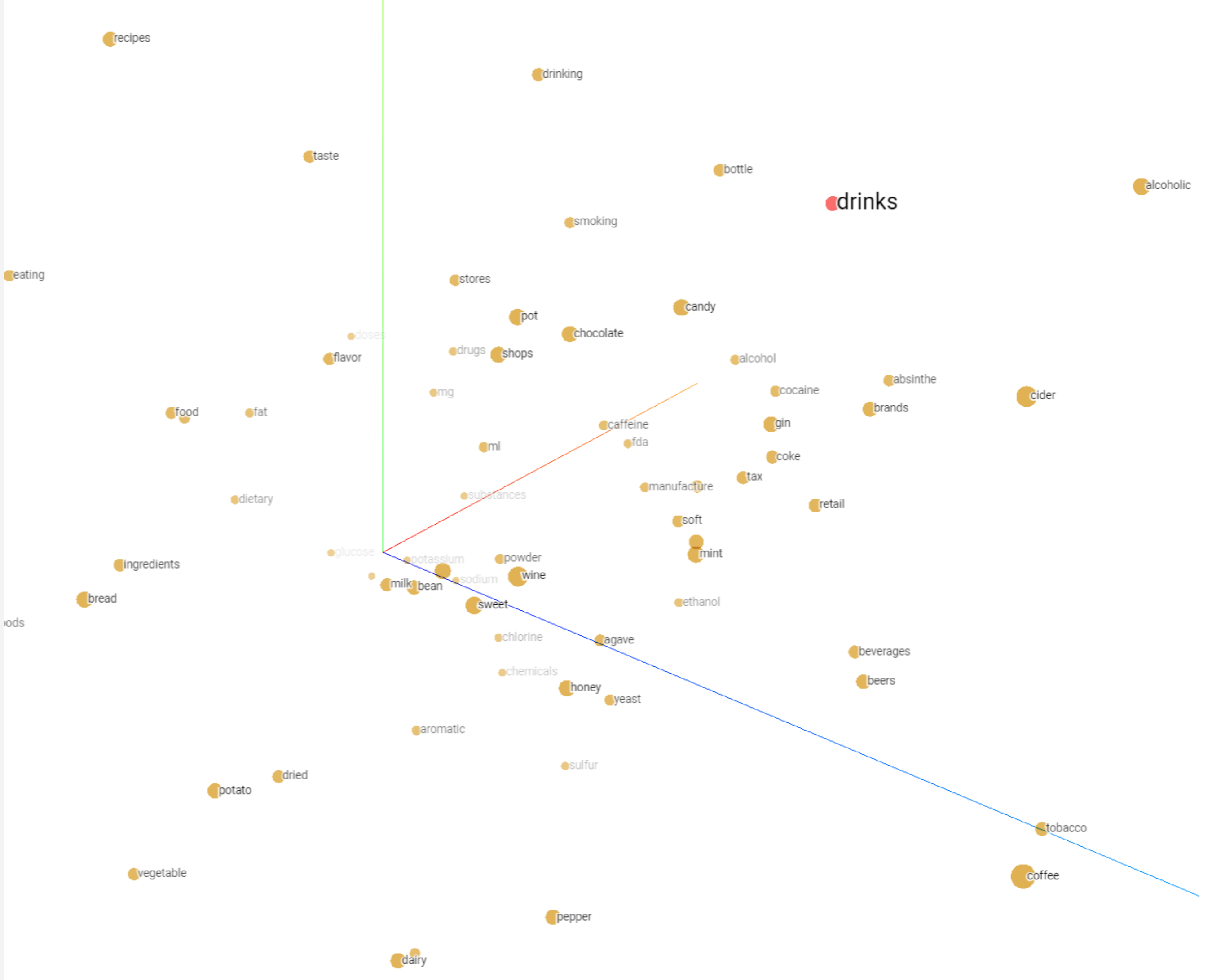
Model (bi-encoder)	Input types	Dimensions
OpenAI: text-embedding-3-small	Text	1536
OpenAI: text-embedding-3-large	Text	3072
Mistral: e5-mistral-7b-instruct	Text	4096

What should we care about vectors?

Search & Similarity

Search and retrieve items that are similar to what you’re querying.

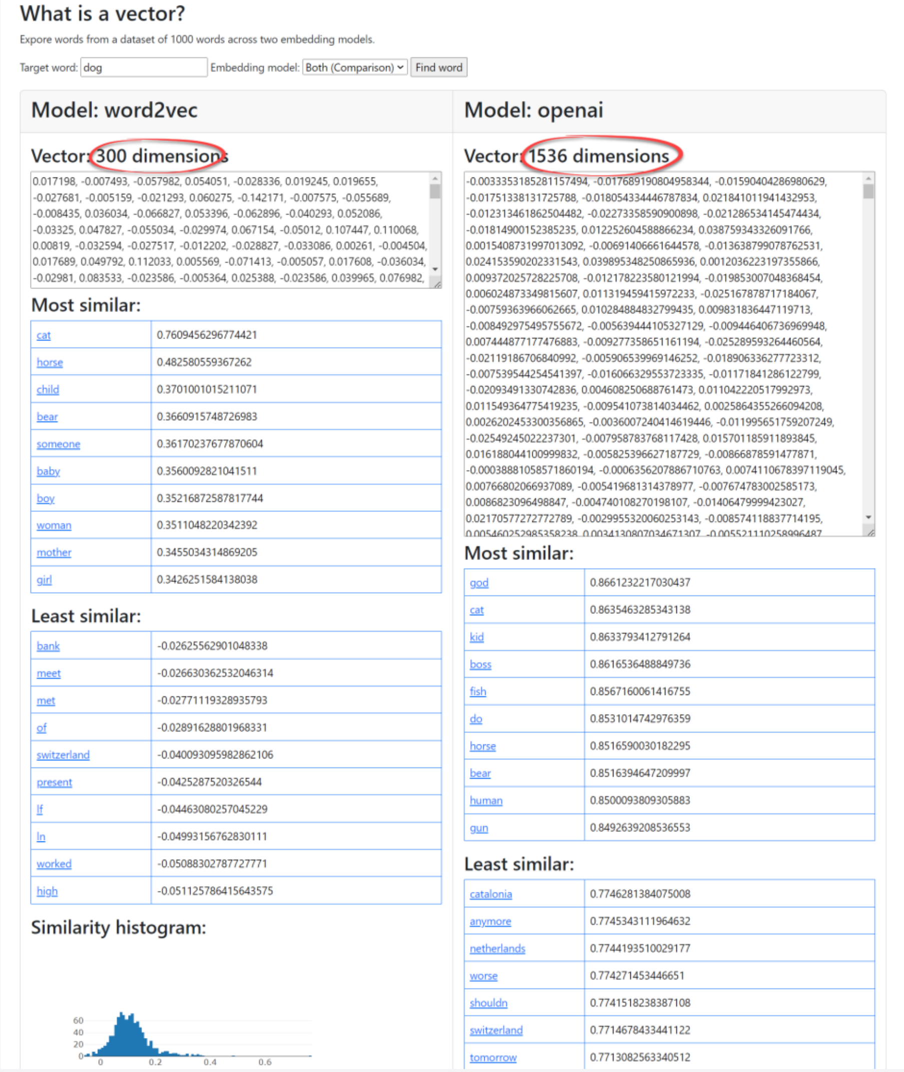
https://projector.tensorflow.org/



Exploring Vectors

Generate Example Vectors

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



What is a vector?

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

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

Model: word2vec

Vector: 300 dimensions

```
0.017198, -0.007493, -0.057982, 0.054051, -0.028336, 0.019245, 0.019655,
-0.027681, -0.005159, -0.021293, 0.060275, -0.142171, -0.007575, -0.055689,
-0.008435, 0.036034, -0.066827, 0.053396, -0.062896, -0.040293, 0.052086,
-0.03325, 0.047827, -0.055034, -0.029974, 0.067154, -0.05012, 0.107447, 0.110068,
0.00819, -0.032594, -0.027517, -0.012202, -0.028827, -0.033086, 0.00261, -0.004504,
0.017689, 0.049792, 0.112033, 0.005569, -0.071413, -0.005057, 0.017608, -0.036034,
-0.02981, 0.083533, -0.023586, -0.005364, 0.025388, -0.023586, 0.039965, 0.076982,
```

Most similar:

cat	0.7609456296774421
horse	0.482580559367262
child	0.3701001015211071
bear	0.3660915748726983
someone	0.36170237677870604
baby	0.3560092821041511
boy	0.35216872587817744
woman	0.3511048220342392
mother	0.3455034314869205
girl	0.3426251584138038

Least similar:

bank	-0.02625562901048338
meet	-0.026630362532046314
met	-0.02771119328935793
of	-0.02891628801968331
switzerland	-0.040093095982862106
present	-0.0425287520326544
If	-0.04463080257045229
In	-0.04993156762830111
worked	-0.05088302787727771
high	-0.051125786415643575

Similarity histogram:

Model: openai

Vector: 1536 dimensions

```
-0.0033353185281157494, -0.017689190804958344, -0.01590404286980629,
-0.01751338131725788, -0.018054334446787834, 0.021841011941432953,
-0.012313461862504482, -0.02273358590900898, -0.021286534145474434,
-0.01814900152385235, 0.012252604588866234, 0.038759343326091766,
0.0015408731997013092, -0.00691406661644578, -0.013638799078762531,
0.024153590202331543, 0.039895348250865936, 0.0012036223197355866,
0.009372025728225708, -0.012178223580121994, -0.019853007048368454,
0.006024873349815607, 0.011319459415972233, -0.025167878717184067,
-0.00759363966062665, 0.010284884832799435, 0.009831836447119713,
-0.008492975495755672, -0.005639444105327129, -0.009446406736969948,
0.007444877177476883, -0.009277358651161194, -0.025289593264460564,
-0.02119186706840992, -0.005906539969146252, -0.018906336277723312,
-0.007539544254541397, -0.016066329553723335, -0.01171841286122799,
-0.02093491330742836, 0.004608250688761473, 0.011042220517992973,
0.011549364775419235, -0.009541073814034462, 0.0025864355266094208,
0.0026202453300356865, -0.0036007240414619446, -0.011995651759207249,
-0.02549245022237301, -0.007958783768117428, 0.015701185911893845,
0.016188044100999832, -0.005825396627187729, -0.00866878591477871,
-0.00038881058571860194, -0.0006356207886710763, 0.0074110678397119045,
0.00766802066937089, -0.005419681314378977, -0.007674783002585173,
0.0086823096498847, -0.004740108270198107, -0.01406479999423027,
0.02170577272772789, -0.0029955320060253143, -0.008574118837714195,
0.005460252985358238, 0.0034130807034671307, -0.005521110258996487
```

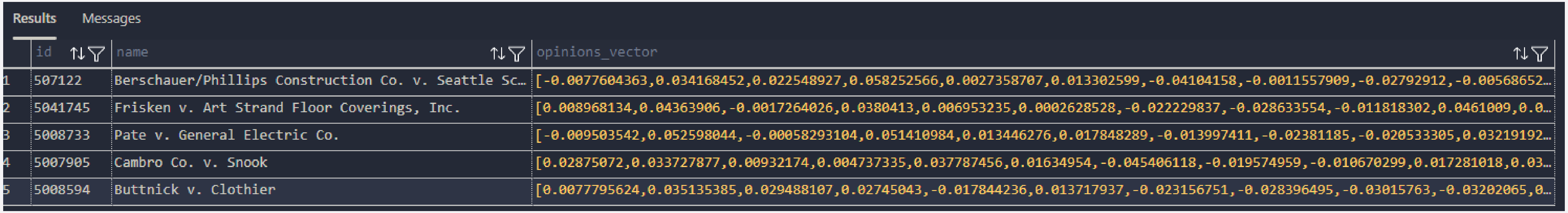
Most similar:

god	0.8661232217030437
cat	0.8635463285343138
kid	0.8633793412791264
boss	0.8616536488849736
fish	0.8567160061416755
do	0.8531014742976359
horse	0.8516590030182295
bear	0.8516394647209997
human	0.8500093809305883
gun	0.8492639208536553

Least similar:

catalonia	0.7746281384075008
anymore	0.7745343111964632
netherlands	0.7744193510029177
worse	0.774271453446651
shouldn	0.7741518238387108
switzerland	0.7714678433441122
tomorrow	0.7713082563340512

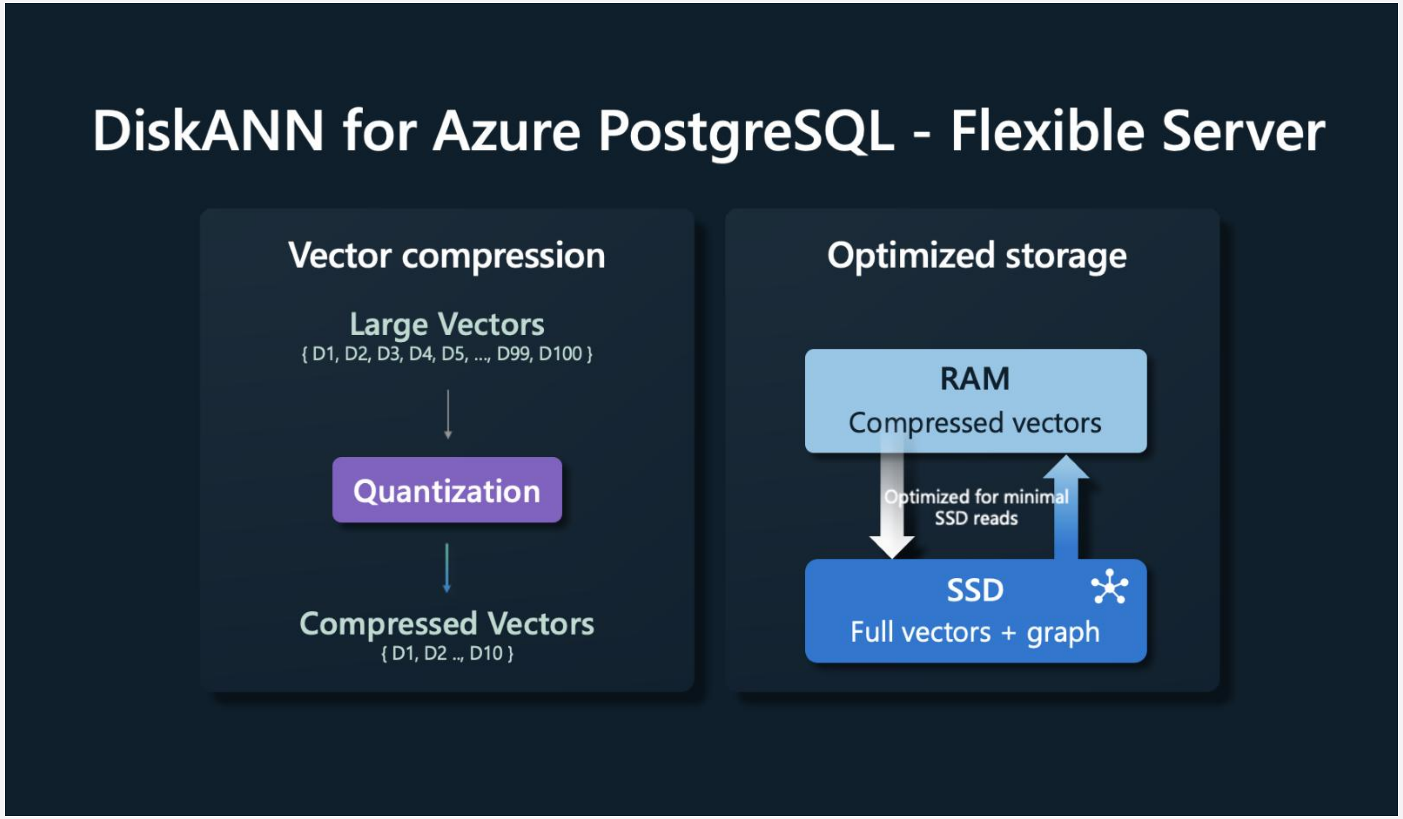
Storing vectors in the PostgreSQL Table



	id	name	opinions_vector
1	507122	Berschauer/Phillips Construction Co. v. Seattle Sc…	[-0.0077604363,0.034168452,0.022548927,0.058252566,0.0027358707,0.013302599,-0.04104158,-0.0011557909,-0.02792912,-0.00568652…
2	5041745	Frisken v. Art Strand Floor Coverings, Inc.	[0.008968134,0.04363906,-0.0017264026,0.0380413,0.006953235,0.0002628528,-0.022229837,-0.028633554,-0.011818302,0.0461009,0.0…
3	5008733	Pate v. General Electric Co.	[-0.009503542,0.052598044,-0.00058293104,0.051410984,0.013446276,0.017848289,-0.013997411,-0.02381185,-0.020533305,0.03219192…
4	5007905	Cambro Co. v. Snook	[0.02875072,0.033727877,0.00932174,0.004737335,0.037787456,0.01634954,-0.045406118,-0.019574959,-0.010670299,0.017281018,0.03…
5	5008594	Buttnick v. Clothier	[0.0077795624,0.035135385,0.029488107,0.02745043,-0.017844236,0.013717937,-0.023156751,-0.028396495,-0.03015763,-0.03202065,0…

Vector Indexing - DiskANN

- Highly performant, scalable, and accurate index for vectors
- Superior to IVFLAT and HNSW
- Reduced memory footprint by storing vectors on SSD
- Compression and quantization improve speed and accuracy of vector search
- Accuracy retained as data changed



Lab “Follow Me”

Lab Part 0 – Login to Azure

In this part of the lab…

1. Log into Azure Portal
2. Verify Azure Services are provisioned

Lab Part 0 – Login to Azure

Azure Services:

ResourceGroup1:

- Azure OpenAI
- Azure PostgreSQL Database

Lab Part 1 – Setup your Azure PostgreSQL Database

(Work from the Lab Manual)

In this part of the lab…

1. Open VS Code
2. Use Connection Dialog to Setup Database Connection
3. Launch PSQL Command Line Shell in VS Code
4. Populate the Database with Sample Data
5. Install and configure the azure_ai extension
6. Explore the azure_ai extension schema
7. Review the Azure OpenAI Schema

Lab Part 2 – Using AI-driven Features in PostgreSQL

(Work from the Lab Manual)

In this part of the lab…

1. Open New Query Editor in VS Code PosgreSQL Extension
2. Using Pattern matching for queries
3. Using Semantic Vector Search and DiskANN Index
   - Create, Store, and Index Embedding Vectors
   - Perform a Semantic Search Query

Lab Part 3 – Build an Agentic App

(Work from the VS Code Notebook)

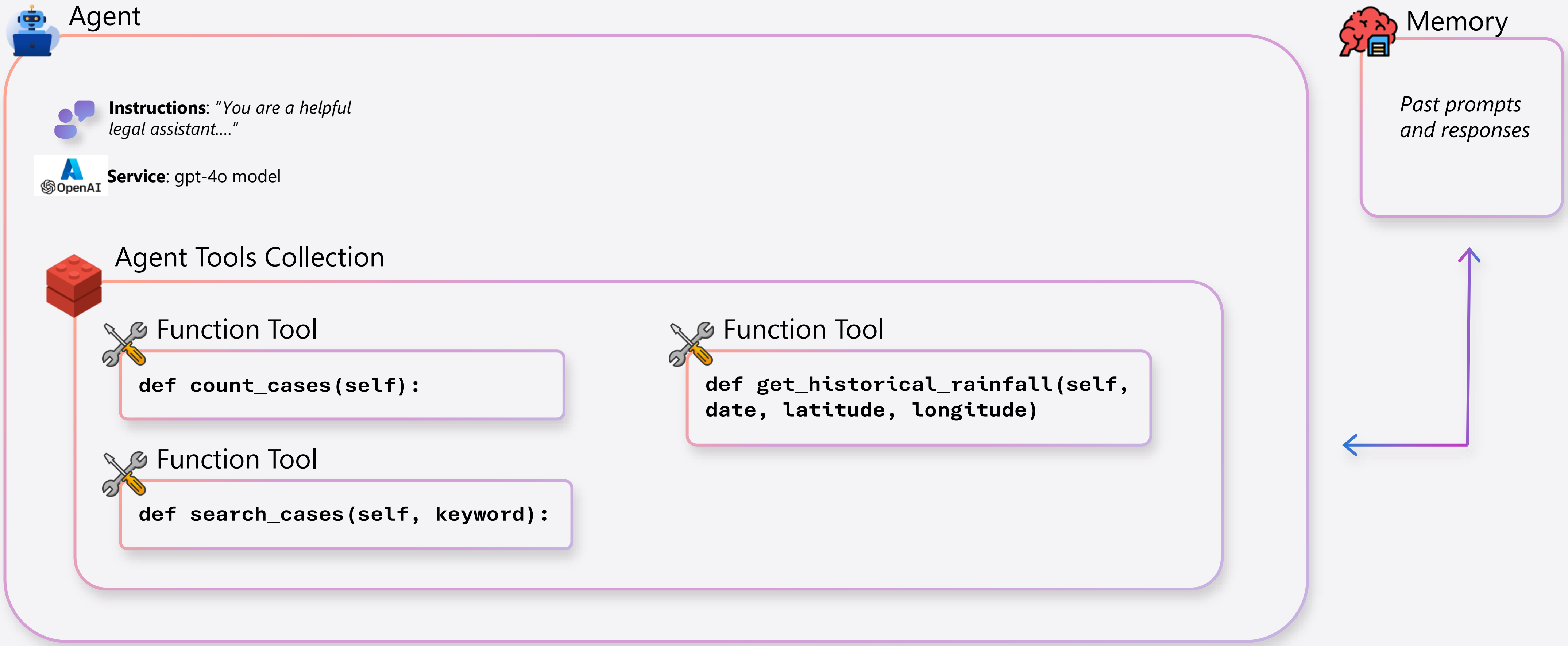
In this part of the lab…

1. Setup Python Imports
2. Setup environmental connection variables
3. Create Agent Framework function for Basic Database Queries
4. Test Run our New Agent
5. Improve Agent Accuracy with Semantic Re-ranking
6. Add GraphRAG Function to Agent
7. Re-assemble our Agent with new GraphRAG Plug-In
8. Add a Weather Service Function
9. Re-Test our Agent with all Functions Together
10. Add memory into the Agent

Additional AI & Dev Concepts

Agents

Agent Framework



Agent

Instructions: *"You are a helpful legal assistant…."*

Service: gpt-4o model

Agent Tools Collection

Function Tool

```
def count_cases(self):
```

Function Tool

```
def search_cases(self, keyword):
```

Function Tool

```
def get_historical_rainfall(self,
date, latitude, longitude)
```

Memory

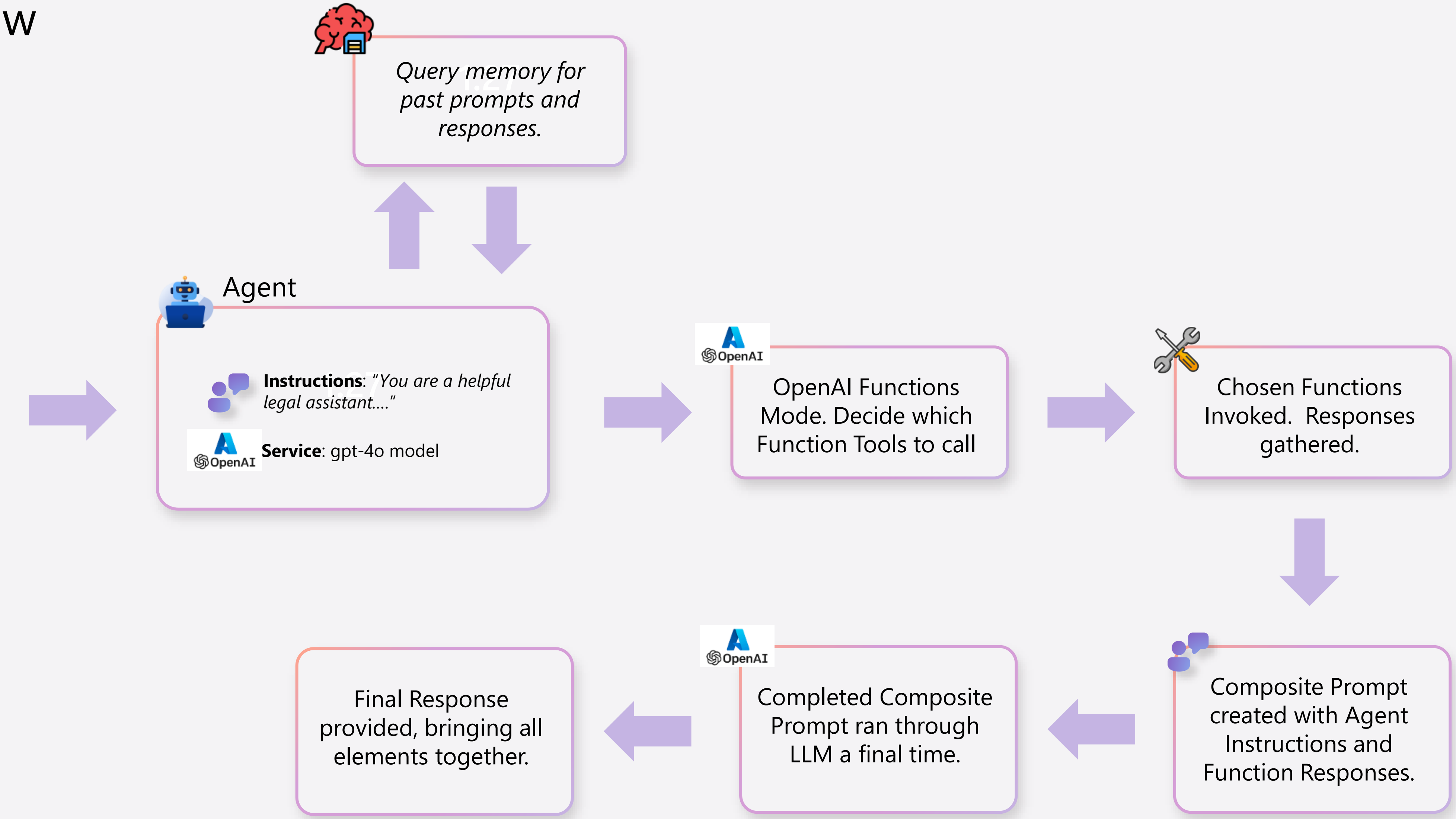
Past prompts and responses

Agents

Logical Flow

Prompt:

"How many cases are there, and high accuracy is important, help me find 5 highly relevant cases related to water leaking in client's apartment."



Agent

Instructions: *"You are a helpful legal assistant…."*

Service: gpt-4o model

Query memory for past prompts and responses.

OpenAI Functions Mode. Decide which Function Tools to call

Chosen Functions Invoked. Responses gathered.

Composite Prompt created with Agent Instructions and Function Responses.

Completed Composite Prompt ran through LLM a final time.

Final Response provided, bringing all elements together.

Semantic Re-Ranking

Process

1. Takes some number (say top 100) vector search results
2. Re-ranks them using cross-encoder model
3. Return top 10 most relevant items

Concepts

- Cross-encoder model performs deeper comparison at text level
- Better relevance on good models
- Requires GPU hardware to run the model

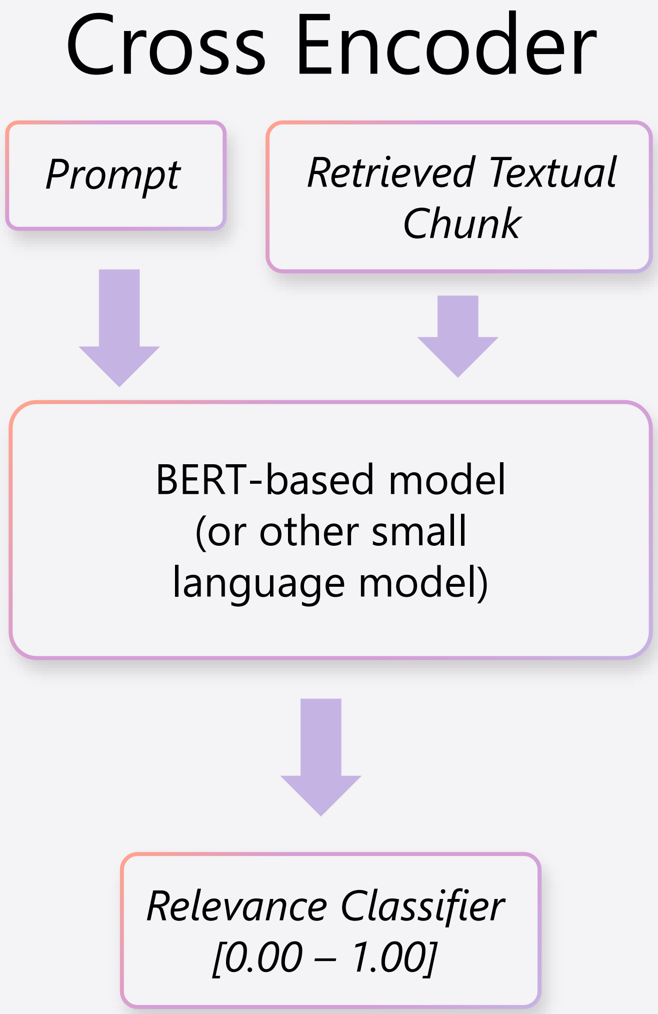
Semantic Re-Ranking

Cross Encoders

- **Process**:
  - A cross-encoder model (e.g., BERT, T5, or Cohere Rerank) compares each retrieved document with the query jointly, considering context from both before ranking.
- **Efficiency**:
  - Higher computational cost, as every document-query pair is encoded dynamically.
- **Example Models**:
  - BGE-reranker-v2-m3, MS MARCO-trained BERT Cross-Encoders, Cohere Rerank Models, T5-based Rerankers

2021 was a major year for efficiency improvements, making them more viable at scale.

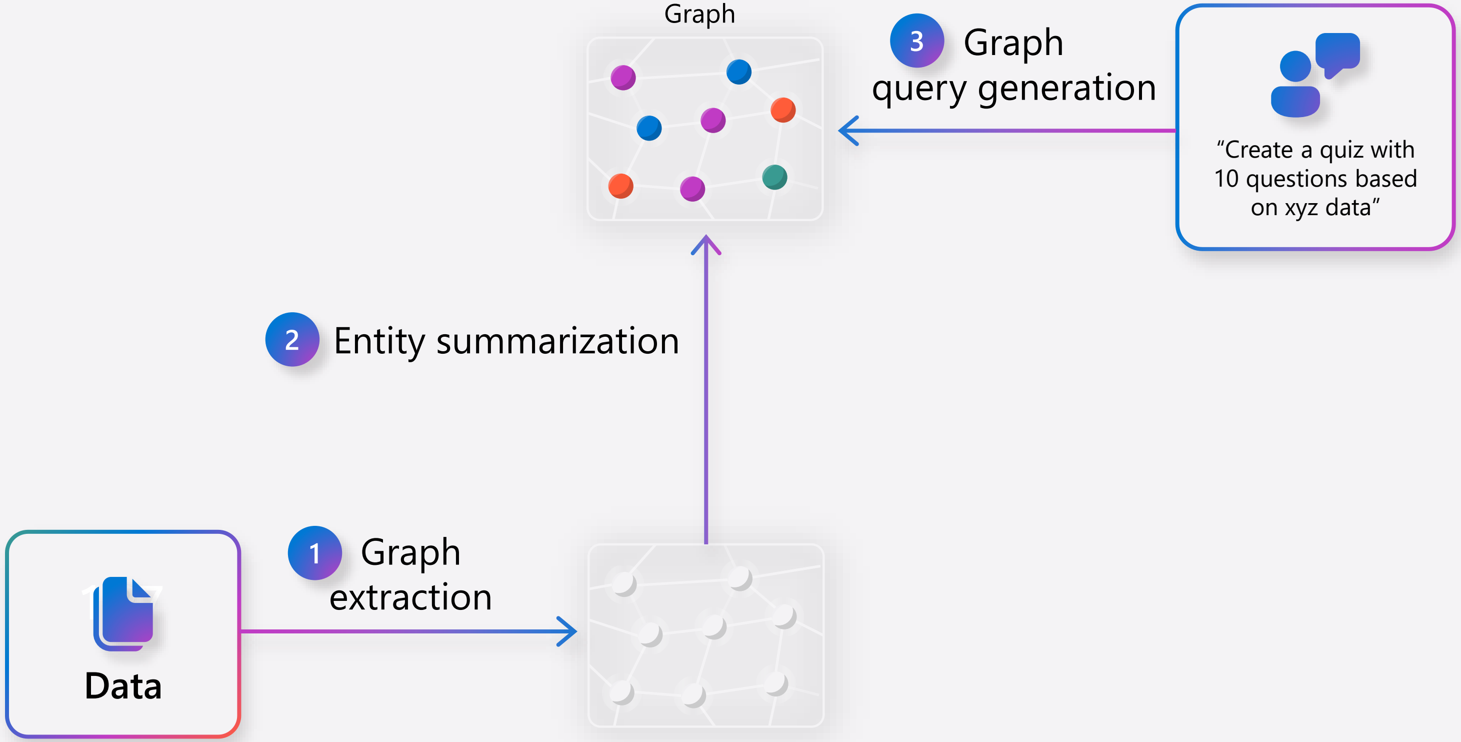
Key papers:
ColBERT (2020) – Khattab & Zaharia, MonoBERT & DuoBERT (2020) – MacAvaney et al., TAS-B (2021) – Hofstätter et al., ColBERTv2 (2021) – Santhanam et al.


Cross Encoder
Prompt
Retrieved Textual Chunk
BERT-based model (or other small language model)
Relevance Classifier [0.00 – 1.00]

GraphRAG – Option 1

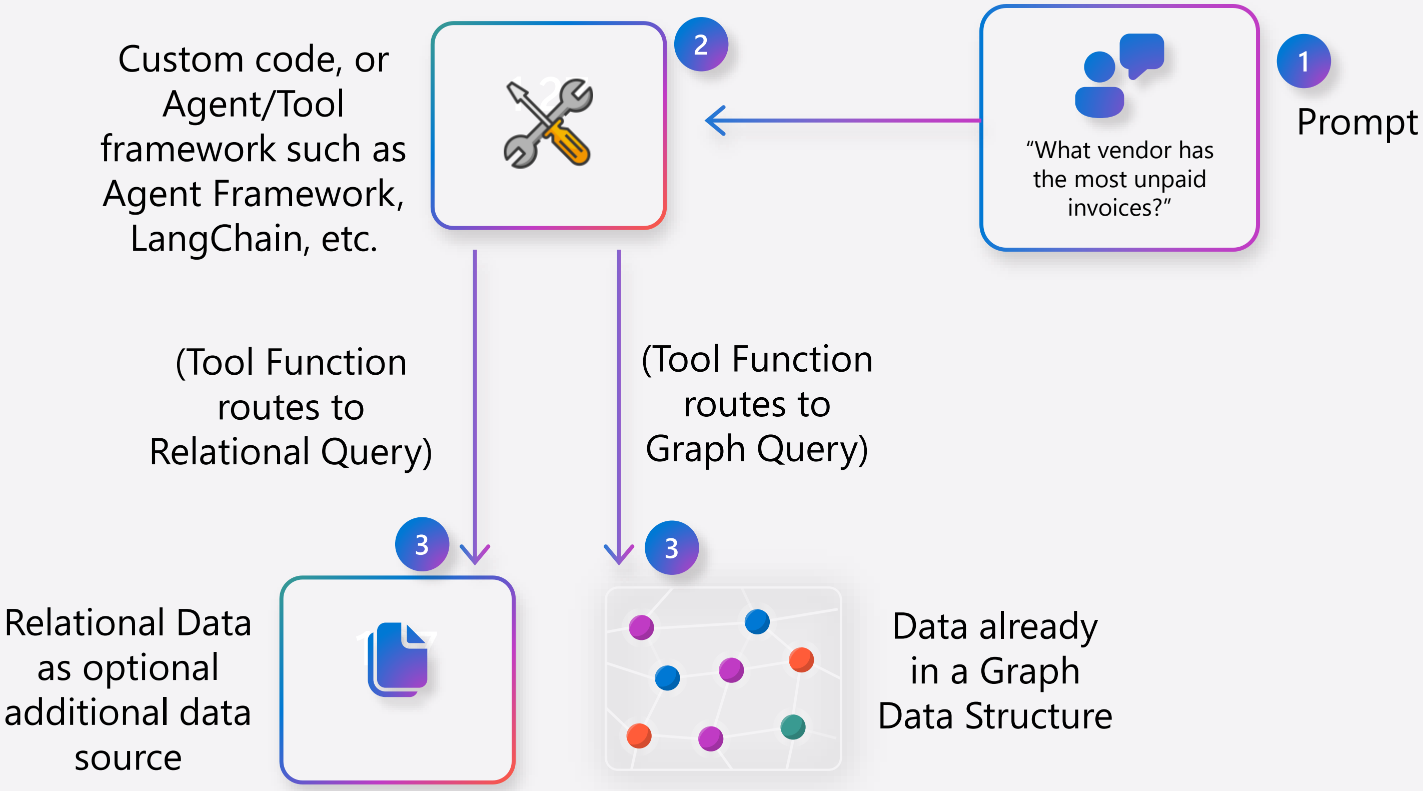
GraphRAG via Post-Processing Graph Construction

(Knowledge Graph Generation)



GraphRAG – Option 2

GraphRAG via Native Graph Data Querying



Custom code, or Agent/Tool framework such as Agent Framework, LangChain, etc.

“What vendor has the most unpaid invoices?”

(Tool Function routes to Relational Query)

(Tool Function routes to Graph Query)

Relational Data as optional additional data source

Data already in a Graph Data Structure

Wrap Up

Related Sessions

BRK130	The blueprint for intelligent AI agents backed by PostgreSQL	**Date**: Wednesday, November 19 **Time**: 11:30 AM - 12:15 PM **Location**: Moscone West, Level 3, Room 3005
BRK123	AI-Assisted Migration: The Path to Powerful Performance on PostgreSQL	**Date**: Thursday, November 20 **Time**: 8:30 AM - 9:15 PM **Location**: Moscone West, Level 3, Room 3003

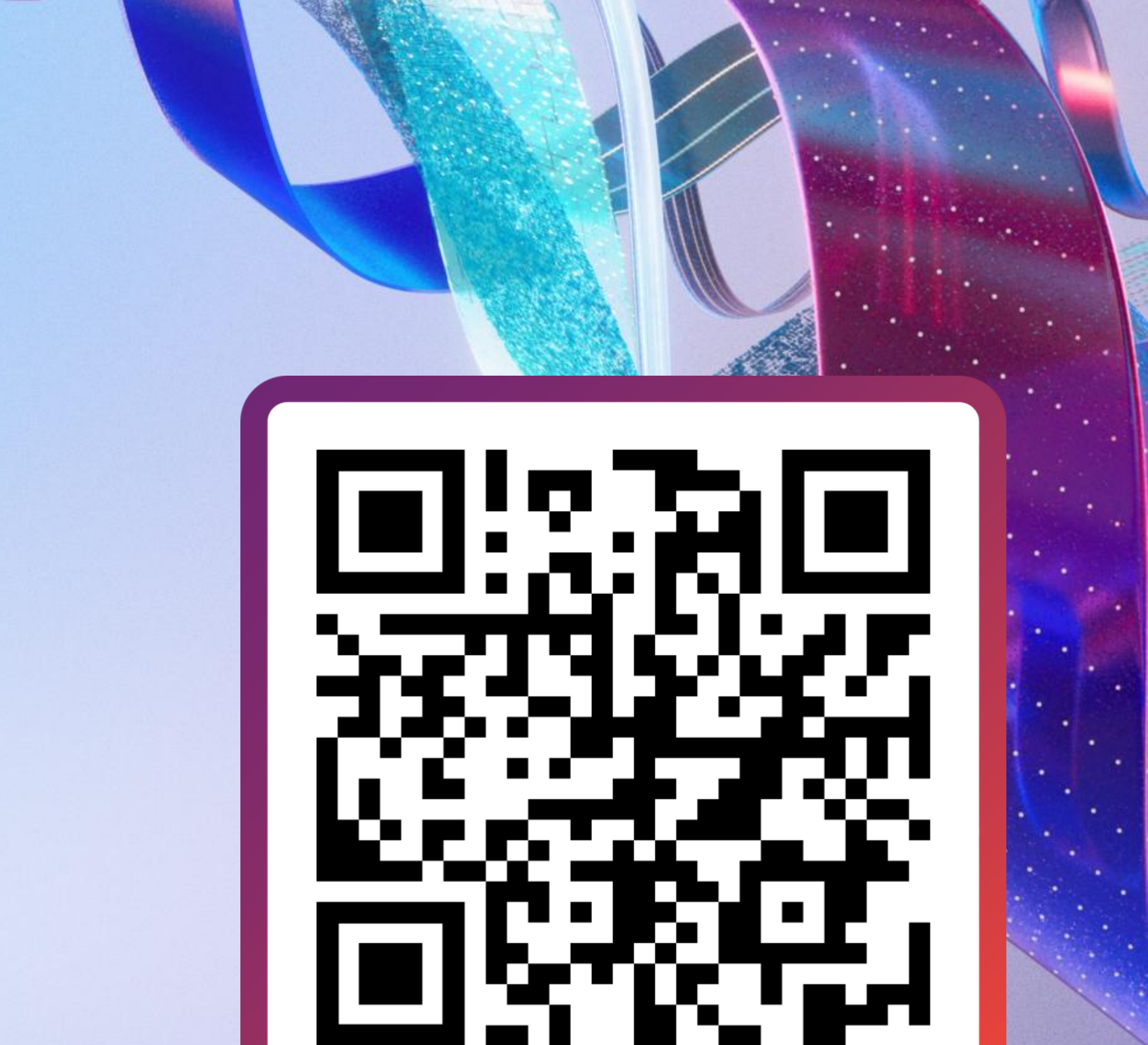
Lab GitHub Repo

Work on the lab later or at home

https://github.com/jjfrost/pg-af-agents-lab

How did we do?

Tell us your thoughts about our sessions and complete the event survey



Go to aka.ms/ignite/sessions/evals or scan the QR code

Get started and build Azure skills on MS Learn

Thank you!