

Applied Deep
Learning

5350 Retrieval Augmented Generation (RAG)

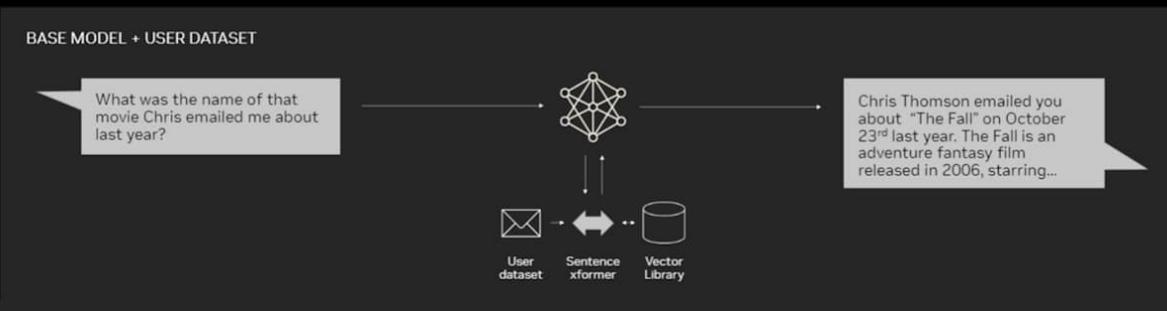
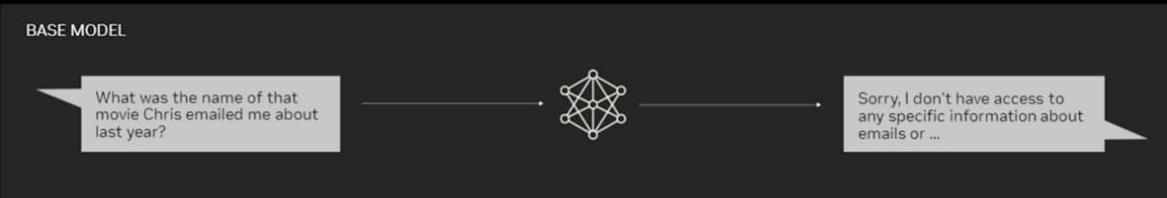
portilla@gmail.com

Jesus.portilla@Colorado.edu

<https://github.com/iportilla/RAG>

Retrieval Augmented Generation (RAG)

USING LOCAL LLMS WITH LOCAL DATA



<https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/>

Business Example (copilot)

Today

What did Santangelo said about his contact information?

 Copilot

Michael Santangelo shared his contact information in an email yesterday morning. He expressed his gratitude for the support he received throughout his career and mentioned that he treasures the relationships built over the years. He provided his contact details as follows ①:

- Email: [msant...](#)
- Office: (720) ...
- Mobile: (303) ...

He also mentioned that he hopes the connections formed will remain and thanked everyone for their friendship ①.

Is there anything else you would like to know about Michael Santangelo?

 **Farewell and Thank You**
Sent yesterday 10:56 AM
 Michael Santangelo to: You; David Ward; + 79 others

1 |  Farewell and Thank You

 **Edit in Pages**     AI-generated content may be incorrect

<https://support.microsoft.com/en-us/copilot-outlook>

LLM Shortcomings

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

```
python                                         Copy

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

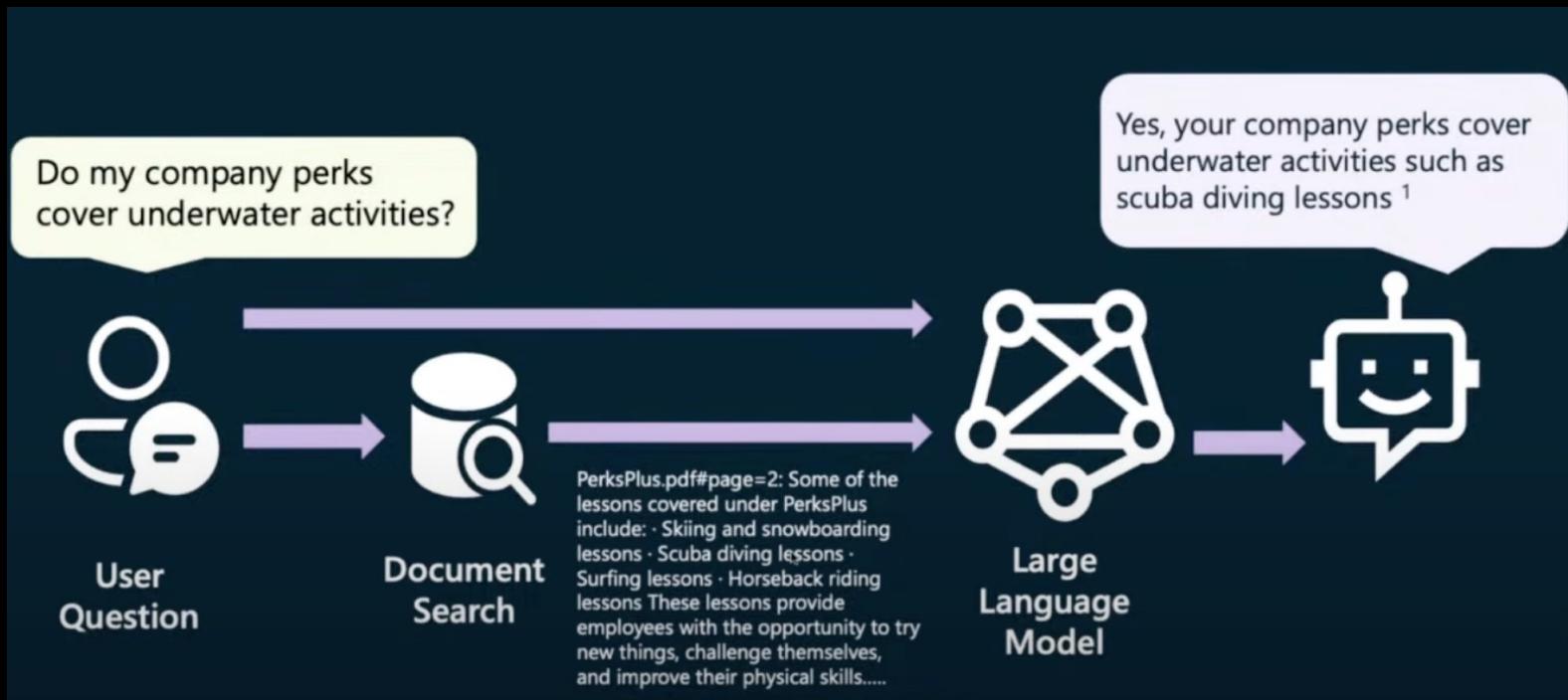
Learn new facts
(temporarily)

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

The Benefits of RAG

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

RAG – Retrieval Augmented Generation



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"

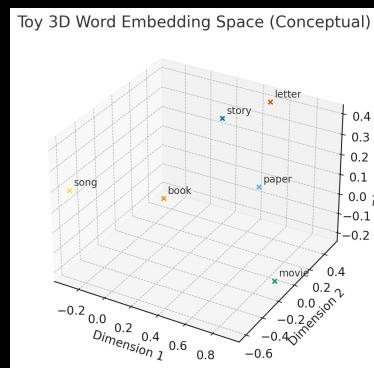
Won't match:

"Scuba classes"

"Snorkeling group sessions"

<https://github.com/iportilla/RAG>

Vector embeddings Lab



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://pamelaf.ox.github.io/vectors-comparison/>

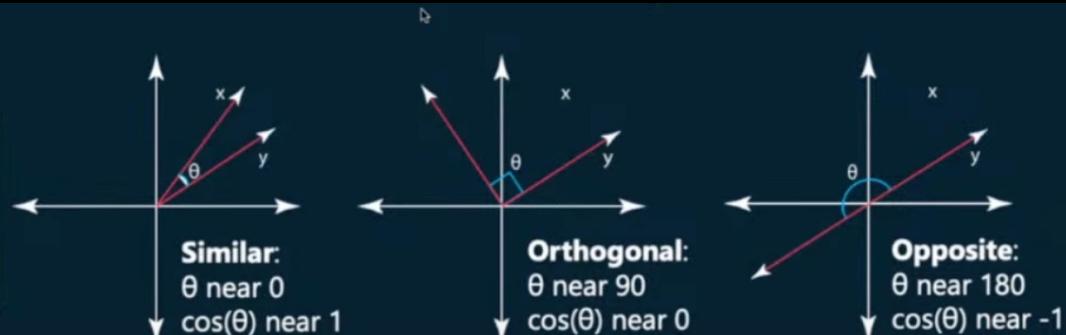
<https://pamelaf.ox.github.io/vectors-comparison/movies.html>

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))
```

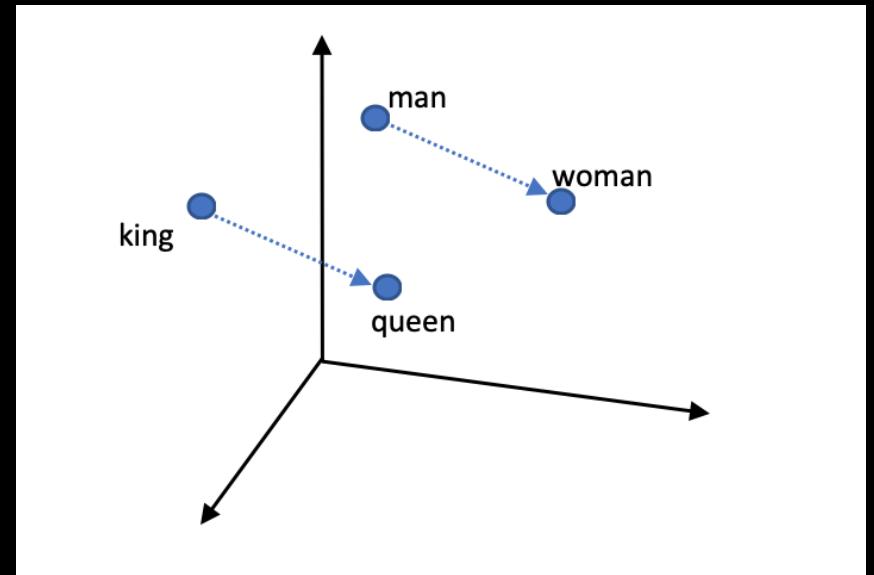
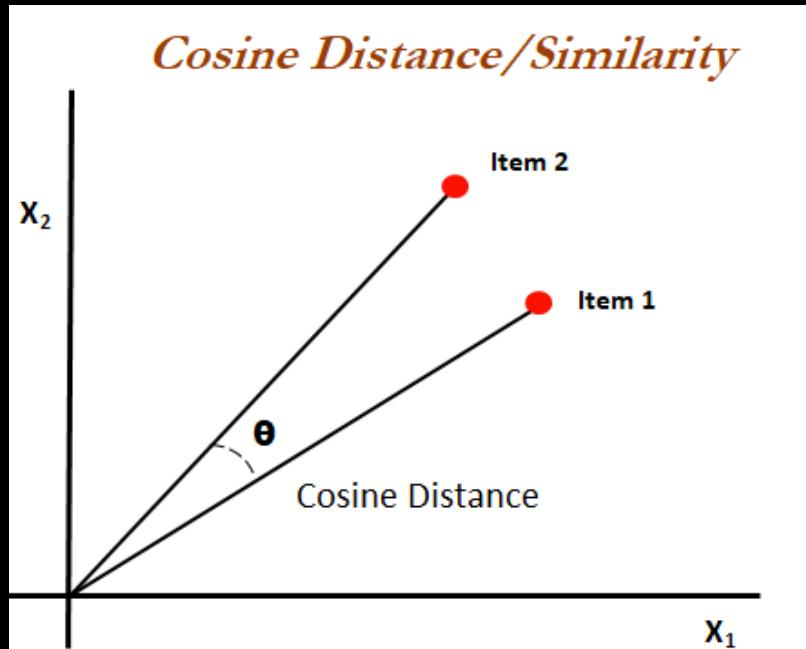


*For ada-002, $\cos(\theta)$ values range from 0.7-1

<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?

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

Target word:

Embedding model: Both (Comparison)

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
film	0.35940013889130784
letter	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://pamelaf.ox.github.io/vectors-comparison/>

Vector Comparison (lab)

What is a vector?

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

Target word: Embedding model: Find word

Model: word2vec	Model: openai																				
Vector: 300 dimensions	Vector: 1536 dimensions																				
0.044865, -0.010391, -0.017868, 0.027773, 0.055985, 0.015209, -0.017383, 0.097498, 0.004765, -0.020102, 0.092006, -0.023716, 0.053701, 0.019798, -0.057878, 0.022918, 0.002671, -0.022792, 0.052438, -0.100994, 0.057101, 0.055935, -0.014476, -0.05498, -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, 0.010729, -0.000422, 0.000279, 0.125454, -0.052438	-0.008843345705419779, -0.019184302053683347, -0.02266409958932397, -0.004917495418339968, -0.02266409958932397,																				
Most similar:	Most similar:																				
<table border="1"><tbody><tr><td>read</td><td>0.3893648604097623</td></tr><tr><td>paper</td><td>0.3634623893904801</td></tr><tr><td>write</td><td>0.35940013889130784</td></tr></tbody></table>	read	0.3893648604097623	paper	0.3634623893904801	write	0.35940013889130784	<table border="1"><tbody><tr><td>paper</td><td>0.8874017308879492</td></tr><tr><td>movie</td><td>0.8805337935966647</td></tr><tr><td>film</td><td>0.8711653176455576</td></tr><tr><td>letter</td><td>0.8632871648170634</td></tr><tr><td>record</td><td>0.8630170946356468</td></tr><tr><td>course</td><td>0.8629488396382509</td></tr><tr><td>bank</td><td>0.8628000814561154</td></tr></tbody></table>	paper	0.8874017308879492	movie	0.8805337935966647	film	0.8711653176455576	letter	0.8632871648170634	record	0.8630170946356468	course	0.8629488396382509	bank	0.8628000814561154
read	0.3893648604097623																				
paper	0.3634623893904801																				
write	0.35940013889130784																				
paper	0.8874017308879492																				
movie	0.8805337935966647																				
film	0.8711653176455576																				
letter	0.8632871648170634																				
record	0.8630170946356468																				
course	0.8629488396382509																				
bank	0.8628000814561154																				

- Walk
- Sleep
- Eat
- Love
- Write
- Read
- Dream
- Run

<https://pamelaf.ox.github.io/vectors-comparison/>

Movie title embeddings in OpenAI

Movie title embeddings in OpenAI

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

Select a movie title:

[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://pamelaf.ox.github.io/vectors-comparison/movies.html>

Movie title embeddings in OpenAI (lab)

Movie title embeddings in OpenAI

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

Select a movie title:

Movie title: The Jungle Book

Vector: 1536 dimensions

-0.009433940052986145, -0.0026398864574730396, 0.002852880861610174, -0.000691843044784462,
-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

Snow White and the Seven Dwarfs

Pinocchio

Fantasia

Song of the South

Cinderella

20,000 Leagues Under the Sea

Lady and the Tramp

Sleeping Beauty

101 Dalmatians

The Absent Minded Professor

Babes in Toyland

Bon Voyage!

The Sword in the Stone

The Jungle Book

Blackbeard's Ghost

The Aristocats

The Boatniks

Bedknobs and Broomsticks

The Apple Dumpling Gang

Freaky Friday

The Many Adventures of Winnie the Pooh

The Rescuers

Herbie Goes to Monte Carlo

The Black Hole

Aspen Extreme

Homeward Bound: The Incredible Journey

The Cemetery Club

Swing Kids

A Far Off Place

Born Yesterday

The Adventures of Huck Finn

Bound by Honor

Indian Summer

Super Mario Bros.

Guilty as Sin

Life With Mikey

What's Love Got to Do With It

Son in Law

Hocus Pocus

Another Stakeout

My Boyfriend's Back

Father Hood

The Joy Luck Club

Money for Nothing

The Program

Cool Runnings

The Nightmare Before Christmas

The Three Musketeers

The Rock

The Hunchback of Notre Dame

Phenomenon

Kazaam

Jack

First Kid

The Rich Man's Wife

D3: The Mighty Ducks

The Associate

Ransom

The War at Home

The Preacher's Wife

Evita

Metro

Prefontaine

Shadow Conspiracy

That Darn Cat

Jungle 2 Jungle

The Sixth Man

Grosse Pointe Blank

Romy and Michele's High School Reunion

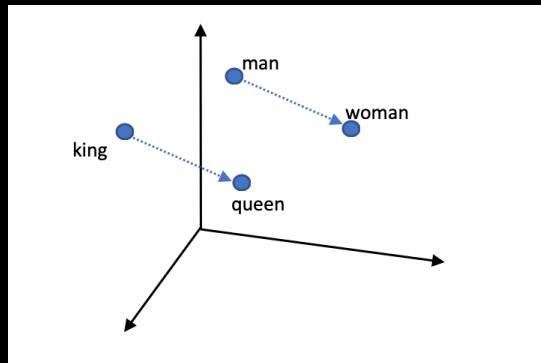
Gone Fishin'

Con Air

Hercules

<https://pamelaf.ox.github.io/vectors-comparison/movies.html>

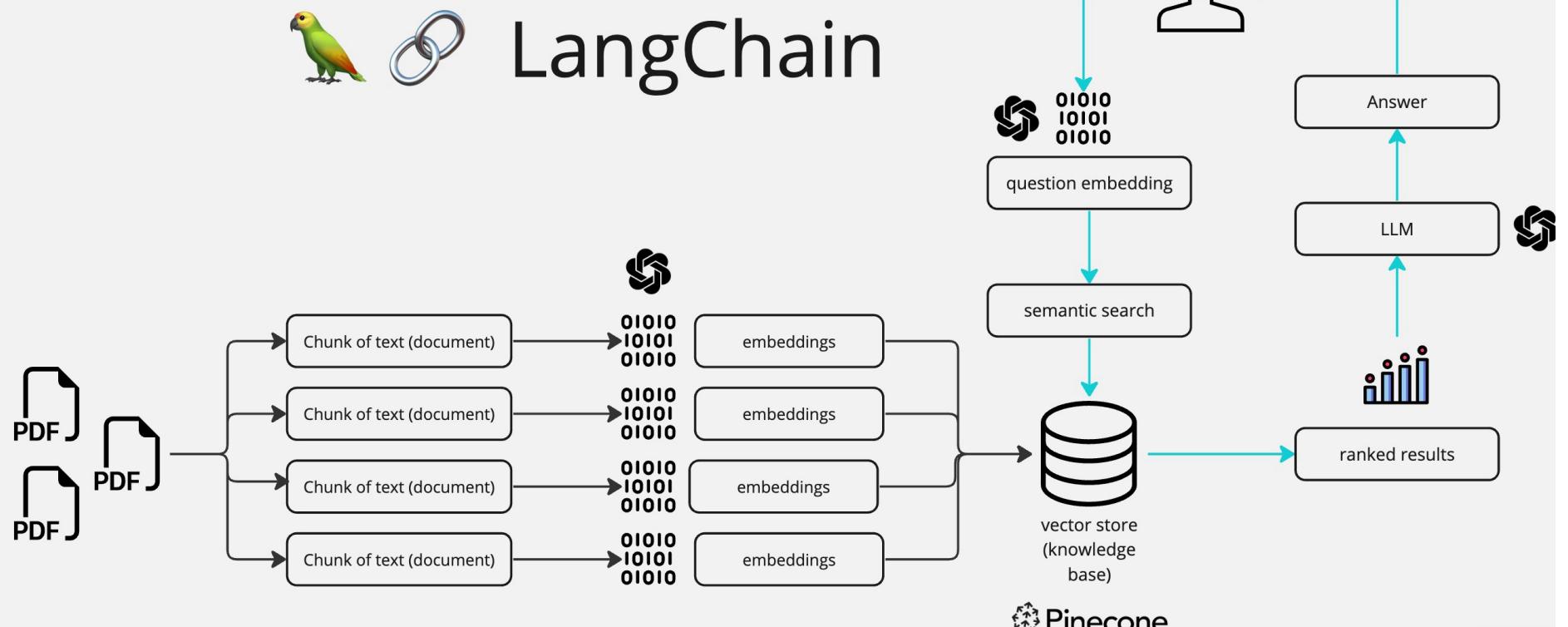
Vector embeddings Lab



<https://aka.ms/aitour/vectors>

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

RAG Lab (search)



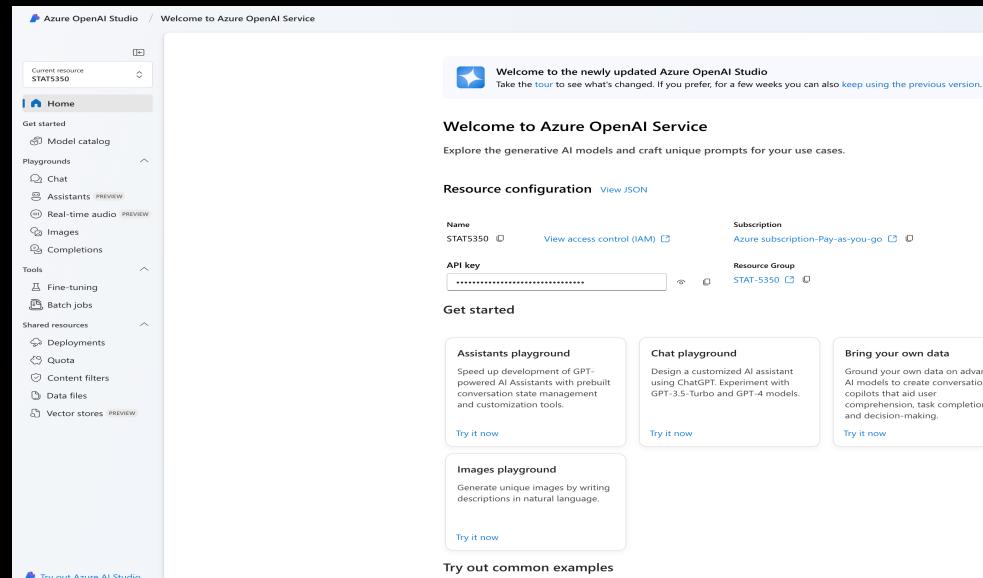
Explain the process to answer this business request:

Leverage Generative AI and sentiment analysis to automatically analyze fine food reviews, extract customer sentiment, and generate insights on product quality, customer preferences, and emerging trends. This will help improve product offerings, tailor marketing strategies, and enhance customer satisfaction in the gourmet food market.

- Azure OpenAI
- RAG
- Exercise

Azure OpenAI

Azure OpenAI Portal.azure.com



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

Azure ML

The screenshot shows the Azure AI | Machine Learning Studio interface. The left sidebar contains navigation links for workspaces, Home, Model catalog, Authoring, Notebooks (selected), Automated ML, Designer, Prompt flow, Assets, Data, Jobs, Components, Pipelines, Environments, Models, Endpoints, Manage, Compute, Monitoring, Data Labeling, Linked Services (PREVIEW), and Connections (PREVIEW). The main area displays a notebook titled "azureml-getting-started-studio.ipynb" under the "Samples" category. The notebook content includes a title "Getting Started: training an image classification model", learning objectives, a list of topics covered, and a section "1. Connect to Azure Machine Learning workspace" with code snippets for handling the workspace.

Notebooks

UCB-O365 > 5350-ML > Notebooks

Files Samples

Search to filter notebooks

azureml-getting-started.ipynb

Want to start editing? Clone this notebook

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](#).

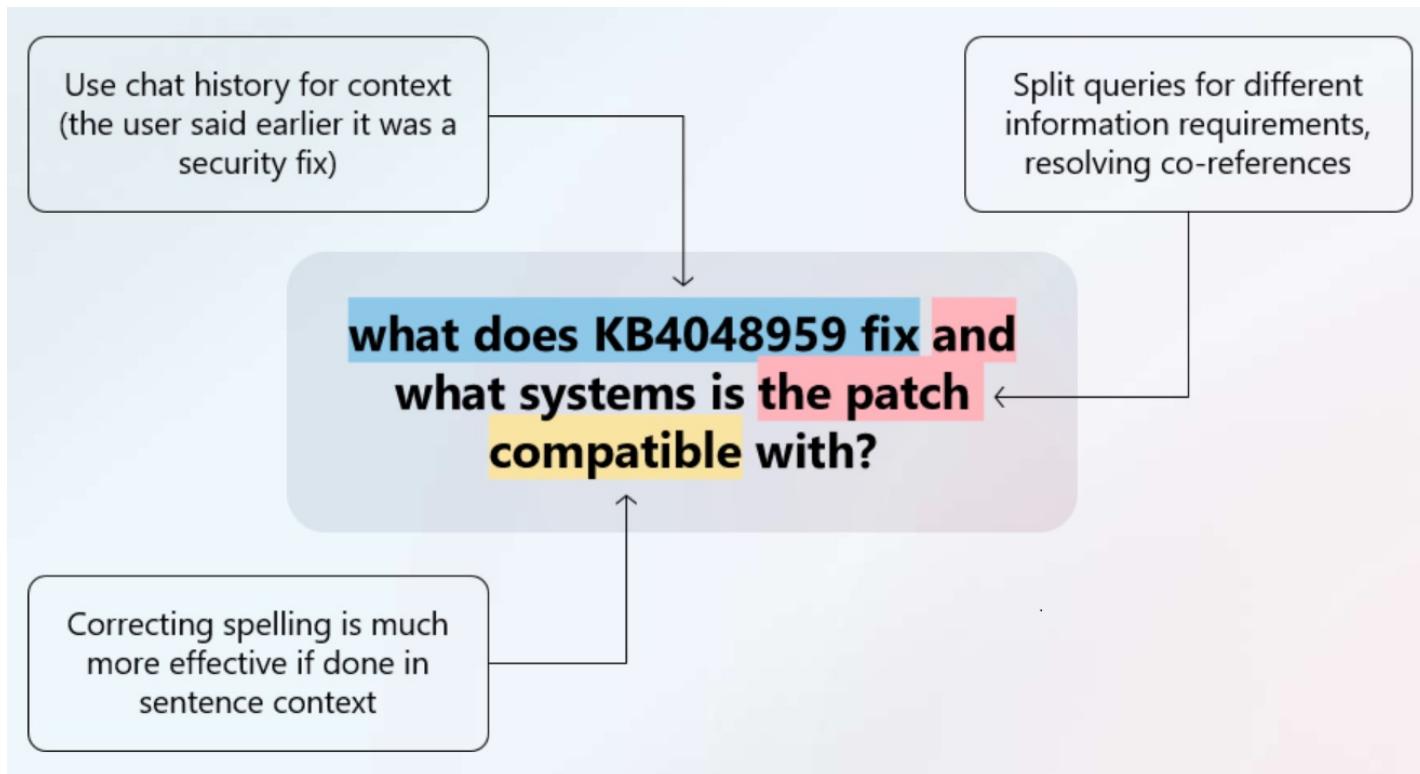
Make sure to enter your workspace credentials before you run the script below.

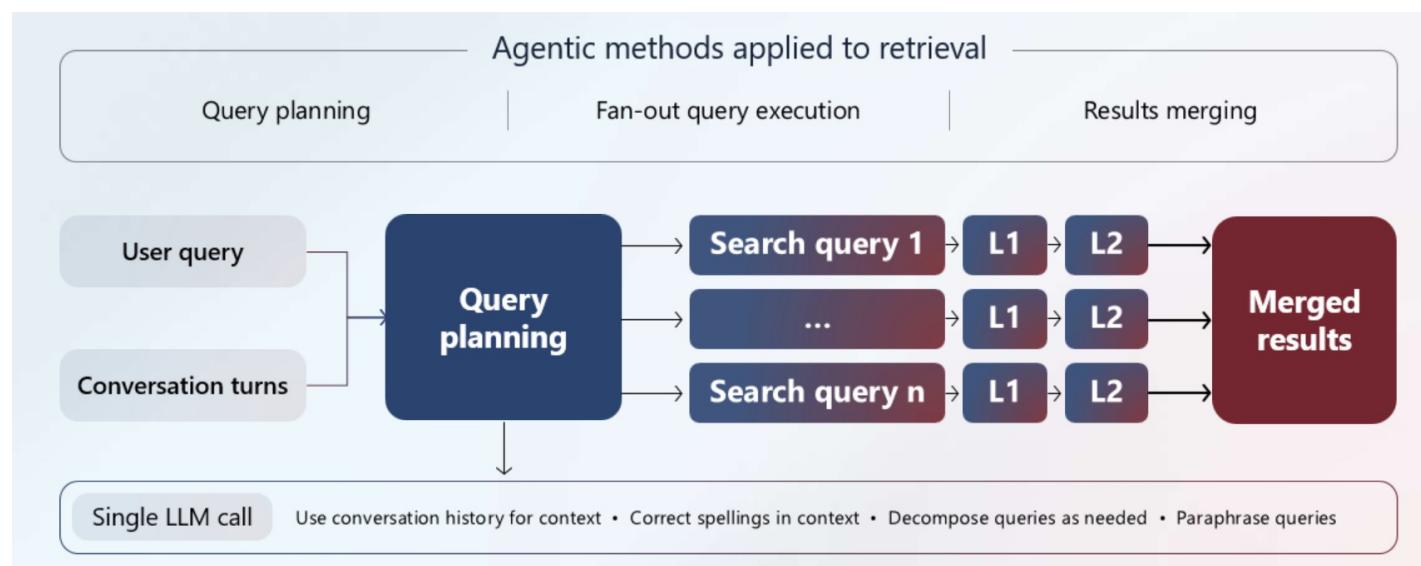
```
1 # Handle to the workspace
2 from azure.ai.ml import MLClient
3
4 # Authentication package
5 from azure.identity import DefaultAzureCredential
6
```

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

Naïve RAG

Example





General Naïve RAG Failure Modes

Weak spot	What goes wrong	Concrete example
Compound / multi-hop questions	A single chunk rarely contains all required facts; similarity search returns pieces that answer only part A or part B, so the LLM hallucinates or guesses.	“Who led the ServiceNow rollout and which cost-saving KPI did the project hit in its first quarter?”
Comparative / option evaluation	Retrieval brings chunks about each item in isolation; the model may produce an ungrounded opinion or miss a dimension.	“Should we host the vector DB on Cosmos DB vs. Pinecone? List trade-offs for latency, price, DevOps effort.”
Long-range summarization & synthesis	Large corpora exceed the context window; sampling yields an incomplete picture.	“Give me the 5-bullet executive summary of all incident-post-mortems from the last fiscal year.”
Numeric aggregation & filtering	Chunks store raw numbers; LLMs struggle with arithmetic over many dispersed values.	“Total downtime minutes for StoreNext clusters across all 2024 incidents—broken down by root cause.”
Temporal reasoning	Vector similarity ignores time; older docs outrank fresher ones unless filtered.	“What are the latest security patches still pending on production?”
Domain-specific jargon / synonyms	Embeddings may not capture subtle synonymy; the right doc stays below top-k.	Query: “steps to fail over the telephony core” → top hits cover PBX firmware updates, but not the DR run-book.

HR Knowledge Base – Naïve RAG Gaps

Weak spot	What goes wrong	Concrete example
Compound / multi-hop questions	Top-k retrieval brings either the policy or the procedure, not both; LLM guesses the missing half.	“How many vacation days can I carry over and which form do I submit to request it?”
Comparative / option evaluation	Chunks describe each benefit plan in isolation; no doc explicitly contrasts them.	“Compare premium costs and coverage between the BlueShield PPO and the Kaiser HMO plans.”
Long-range summarization & synthesis	Engagement-survey results span dozens of PDFs; only a few get stuffed into the prompt.	“Give me a three-bullet summary of the 2024 Employee Engagement Survey across all regions.”

General Naïve RAG Failure Modes vs Agentic RAG

Naïve Weakness	What Goes Wrong	Agentic RAG Fix
Compound / Multi-hop Questions	Only retrieves part of the answer; LLM guesses the rest.	Agent decomposes question and retrieves for each sub-part.
Comparative / Option Evaluation	Retrieves isolated facts; lacks structured comparison.	Agent compares retrieved content explicitly, may use tools.
Long-range Summarization	Incomplete context from large corpora.	Uses map-reduce (Hadoop) summarization across batches of docs.
Numeric	LLMs fail at multi-	Agent calls external

Agentic RAG Fixes – HR Examples

Examples from HR knowledge base:

- Vacation days + form → Agent splits into policy + procedure lookup.
- PPO vs HMO → Agent builds side-by-side comparison.
- Survey summary → Map-reduce summarization over all regions.
- Holiday averages → Uses arithmetic tools to aggregate by country.
- Parental leave after 2023 → Filters results using metadata.

Mitigation techniques

- 1. Iterative (multi-hop) retrieval – decompose the user query into sub-questions; retrieve for each hop; chain the answers.**
- 2. Reranking & fusion-in-decoder – pull a wider candidate set ($k = 50\text{-}100$), rerank with a cross-encoder, then weave snippets together during generation.**
- 3. Hierarchical or map-reduce summarization – summarize batches of chunks, then summarize the summaries.**
- 4. Structured tool calls – push arithmetic, filtering, or comparison to an external tool (SQL, Pandas) and feed results back to the LLM.**
- 5. Metadata-aware retrieval – store and query by tags like “date,” “system,” or “owner,” not just dense vectors.**
- 6. Agentic orchestration – use planners (e.g., ReAct, GraphRAG) that can decide “I need more context for part **R**—run another search”**

Summary: Techniques Used by Agentic RAG

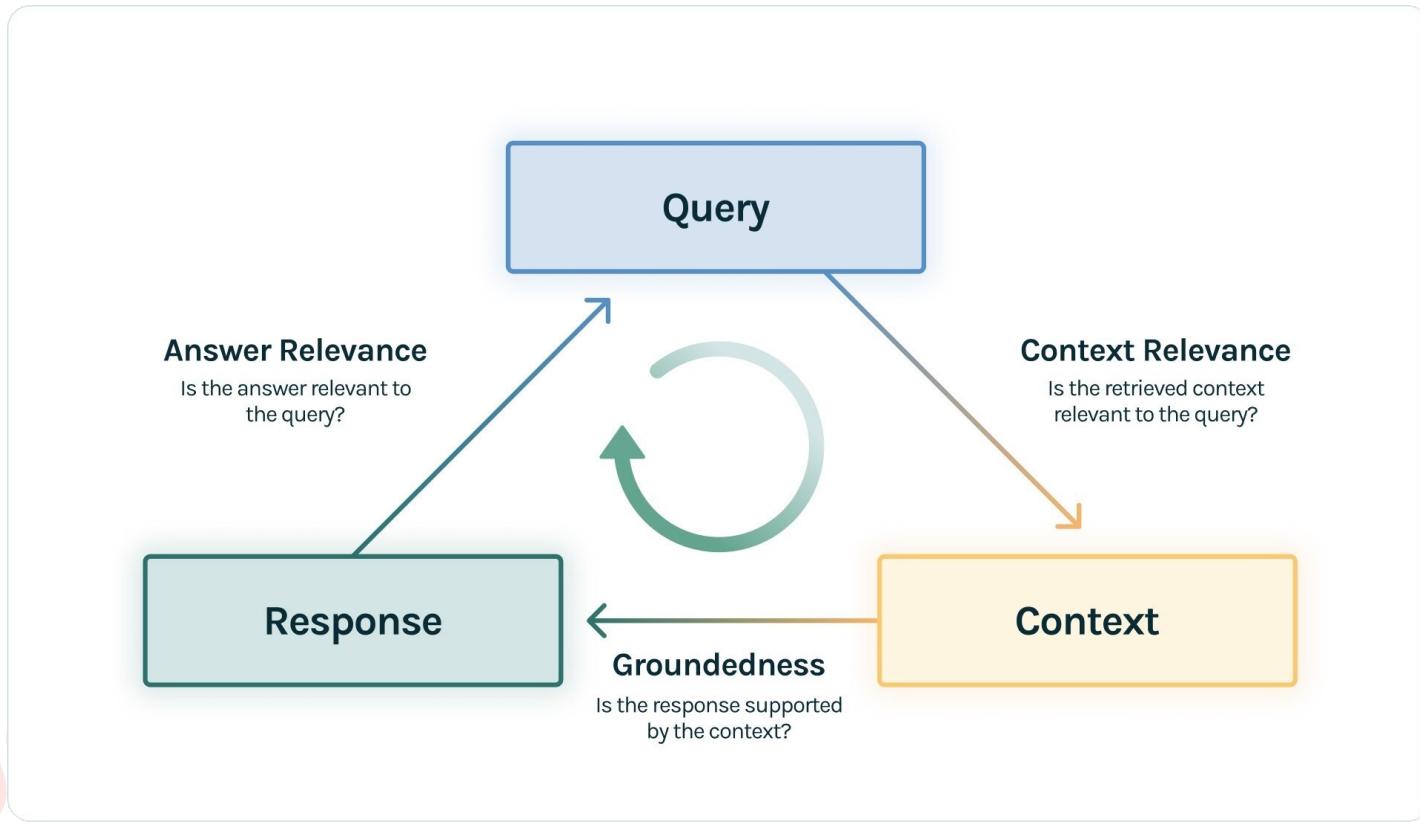
Agentic RAG combines multiple advanced techniques:

- Multi-hop planning with ReAct / GraphRAG
- Metadata filtering (e.g., by date, region)
- Structured tools for math & filtering (SQL, Pandas)
- Hierarchical summarization (map-reduce style)
- Cross-encoder reranking and fusion-in-decoder
- Reflective retry if answer is incomplete

Naïve/Simple RAG vs Agentic RAG

Feature	Naïve RAG	Agentic RAG
Retrieval	Static query to vector DB	Multi-hop, dynamic tool usage
Context handling	Fixed chunks	Context-aware reasoning
Execution	Single-pass	Iterative planning & refinement
Integration	Standalone answer	Workflow automation & action execution

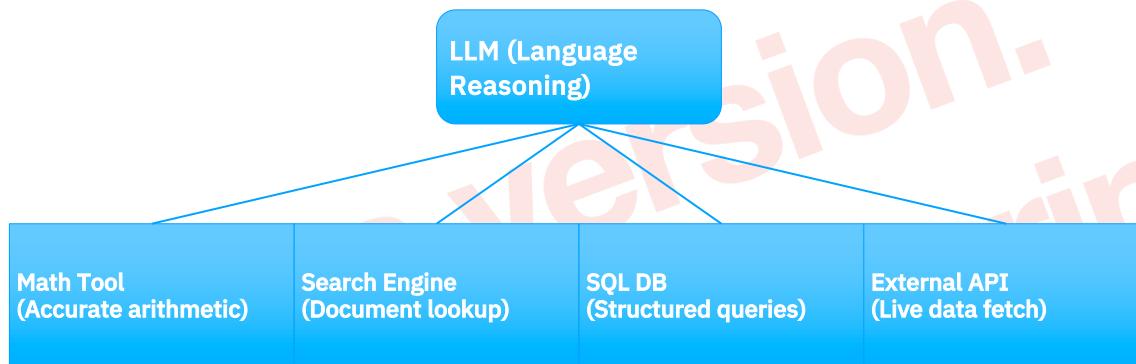
The RAG Triad



https://www.trulens.org/getting_started

IBM

MRKL: Modular Reasoning with Knowledge and Language



The MRKL pattern uses an LLM to route sub-tasks to specialized tools (math, search, SQL, APIs), then synthesizes the outputs into a final, coherent answer.

Key takeaway

- A naïve “**retrieve-top-k-then-stuff**” pipeline is great for *local fact lookup* but brittle for reasoning that spans documents, options, or time.
- Layering smarter retrieval (filters, rerankers), structured reasoning, and multi-step orchestration transforms RAG from a FAQ helper into a robust enterprise knowledge system.

Agentic Retrieval Concepts

Key Insights from Microsoft Learn

Agent = Reasoning + Memory + Action

- Agents do more than retrieve:
- Reason about the user's intent.
- Maintain memory of previous steps and gaps.
- Take action: search, calculate, call tools, refine queries.

Example: For Q2 metrics, the agent retrieves data, runs calculations, and revises the output as needed.

Orchestration Patterns for Agents

- Agentic RAG uses intelligent workflows:
- ReAct: Combines reasoning and tool use (think-act-think).
- MRKL: Modular Reasoning with tools for math, APIs, and search.
- GraphRAG: Uses a knowledge graph for semantic search and traversal.

Structured Retrieval = Higher Precision

- Agents filter and rank with metadata:
- Use fields like doc_type, project_id, or date_updated.
- Improves results for time-sensitive or domain-specific queries.
- Avoids reliance on dense vectors alone.

Example: Filter to docs updated in the last 30 days to answer 'What's still pending?'

Tool Use is First-Class

- Agents don't just guess—they use real tools:
- SQL queries, APIs, Excel-like computation tools.
- Enables accurate math, comparison, and dynamic lookup.
- Supports grounded enterprise logic.

Example: To compare sprint velocity, the agent queries Jira and calculates rankings.

Memory Powers Intelligent Retrieval

- Agents keep track of what's known or missing:
- Maintains short-term memory of retrieved and used documents.
- Prevents repetition, hallucination, and context gaps.
- Enables long-form synthesis across steps.

Agent ≠ Just an LLM

- Agents orchestrate, LLMs assist:
- Agent decides when and how to retrieve.
- LLM generates queries, parses output, or writes summaries.
- Tools and APIs provide facts, calculations, and results.
- Enhances traceability, security, and flexibility.

Final Takeaway

- Agentic RAG = Think + Plan + Act
- Not just Q&A – it's structured knowledge work.
- Dynamically adapts to user needs.
- Uses tools, memory, filters, and multi-step logic.
- Ideal for robust enterprise-scale answers.

Resources

- <https://www.youtube.com/watch?v=Fjef7uhV3gw>
- <https://learn.microsoft.com/en-us/azure/search/search-get-started-rag?pivots=python>
- <https://learn.microsoft.com/en-us/azure/search/search-agentic-retrieval-concept>
- <https://www.youtube.com/watch?v=PeTmOidqHM8>
- https://www.trulens.org/getting_started/core_concepts/rag_triad/
- <https://github.com/Azure-Samples/azure-search-python-samples>