Microsoft

# Microsoft AI Tour

Microsoft

# Vector search and state-of-the-art retrieval for generative AI apps

Pamela Fox
Principal Cloud Advocate (Python)

# Agenda

- Retrieval-augmented generation (RAG)
- Vectors and vector databases
- State of the art retrieval with Azure AI Search
- Data and platform integrations
- Use cases

# Retrieval-augmented generation (RAG)

# The limitations of LLMS

Outdated public knowledge

**You**

Write a model class using latest version of Flask SQL Alchemy

**Assistant**

Sure! Here's an example of a model class using the latest version of Flask SQLAlchemy:

```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class User(db.Model):
    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(50), unique=True)
    email = db.Column(db.String(120), unique=True)
```

**You**

Do my company perks cover underwater activities?

**Assistant**

To determine whether your company perks cover underwater activities, you should consult your employee benefits package or reach out to your human resources department. They will be able to provide you with detailed information regarding the specific perks and benefits offered by your company, including any coverage for underwater activities.

No internal knowledge

# Incorporating domain knowledge

### Prompt engineering

In-context learning

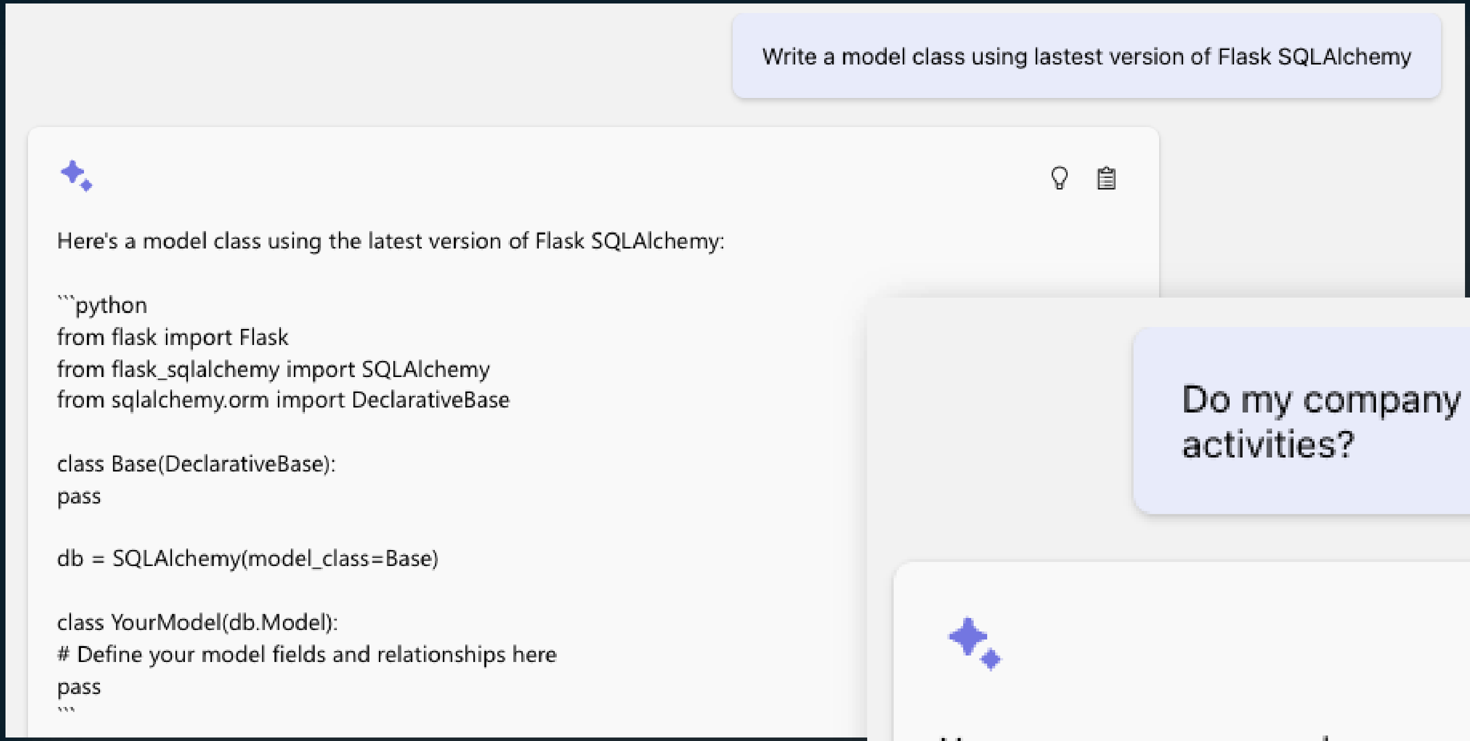
### Fine tuning

Learn new skills (permanently)

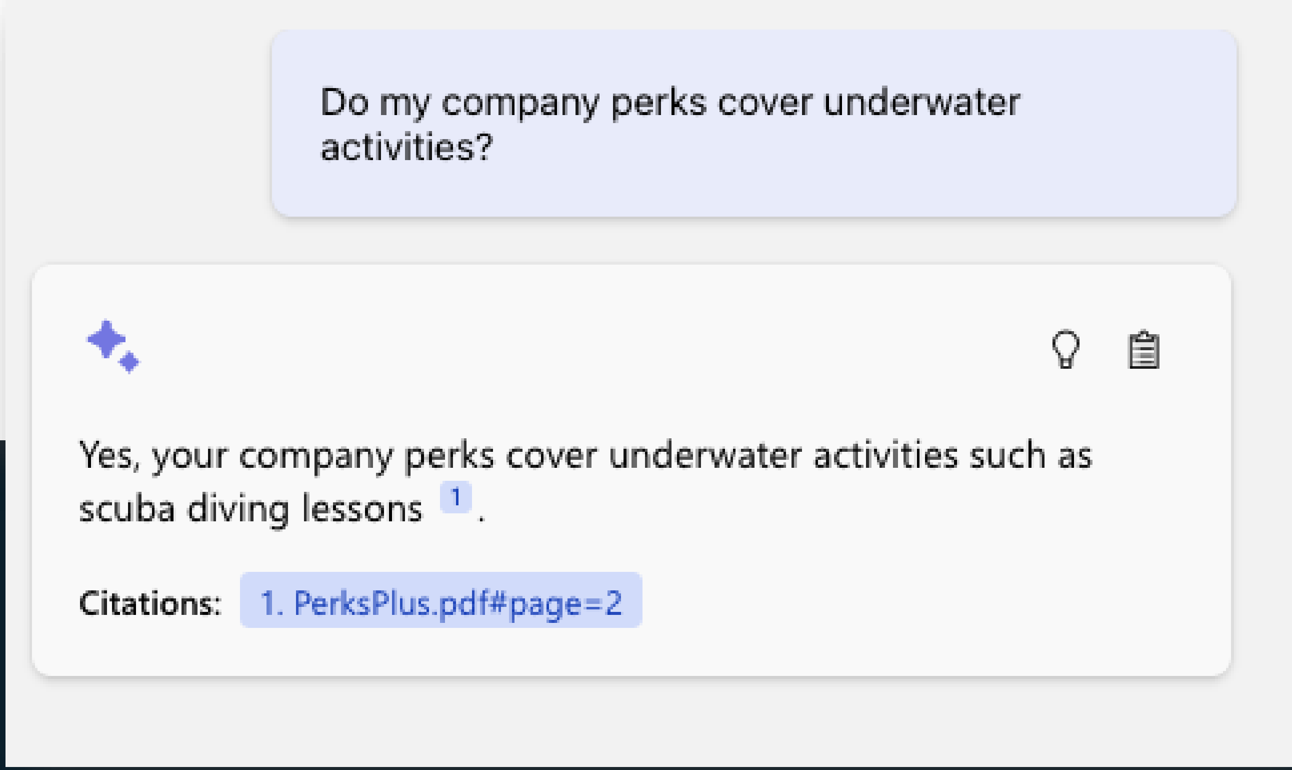
### Retrieval augmentation

Learn new facts (temporarily)

# The benefit of RAG

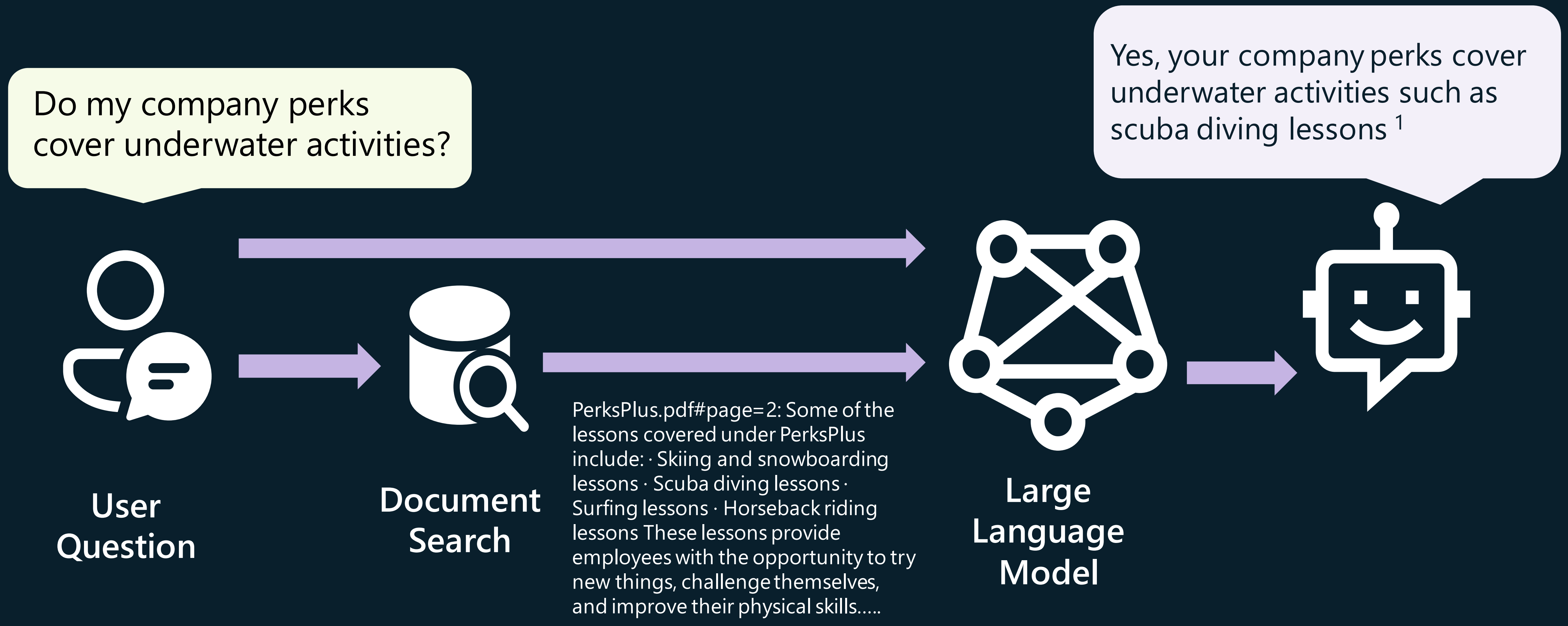



Up-to-date public knowledge




Access to internal knowledge

# RAG – Retrieval Augmented Generation



# 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.)

## Example

**Question:**

“Looking for lessons on underwater activities”

**Won’t match:**

“Scuba classes”

“Snorkeling group sessions”

# Vectors and vector databases

# Vector embeddings

An embedding encodes an input as a list of floating-point numbers.

"dog" → [0.017198, -0.007493, -0.057982, 0.054051, -0.028336, 0.019245,…]

Different models output different embeddings, with varying lengths.

| Model | Encodes | Vector length |
|---|---|---|
| word2vec | words | 300 |
| Sbert (Sentence-Transformers) | text (up to ~400 words) | 768 |
| OpenAI ada-002 | text (up to 8191 tokens) | 1536 |
| Azure Computer Vision | image or text | 1024 |

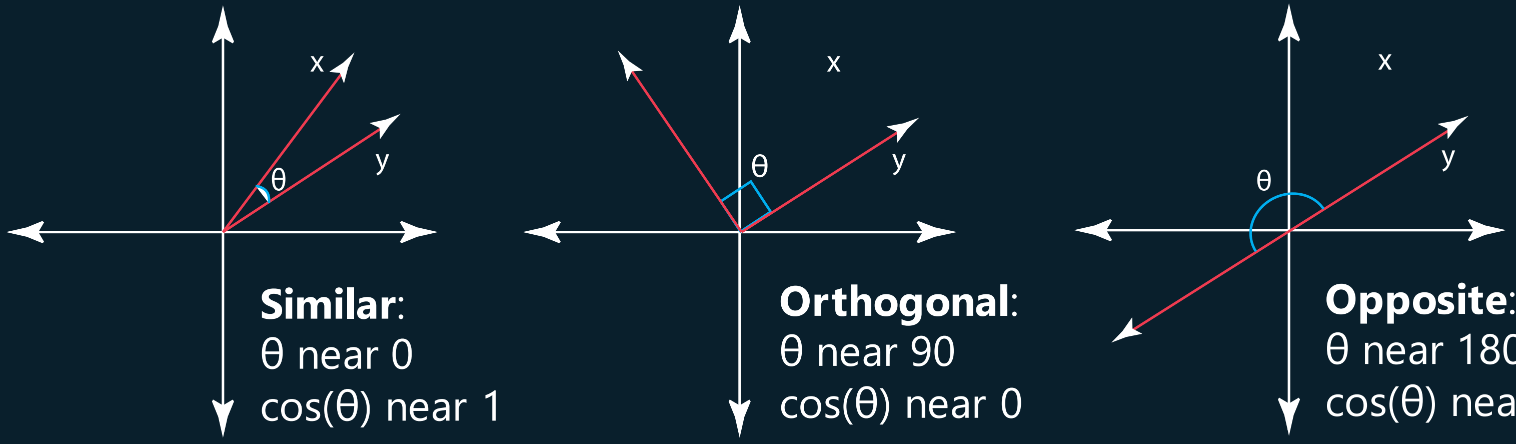
….and many more models!

💻 Demo: Compute a vector with ada-002 (aka.ms/aitour/vectors)

# Vector similarity

We compute embeddings so that we can 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))
```




**Similar**:
θ near 0
cos(θ) near 1

**Orthogonal**:
θ near 90
cos(θ) near 0

**Opposite**:
θ near 180
cos(θ) near -1

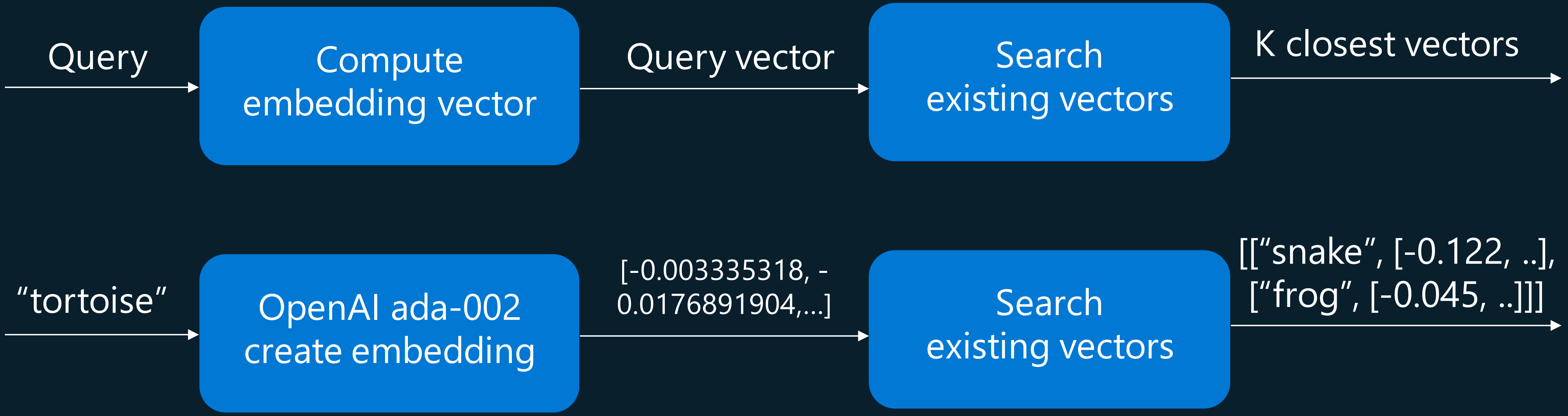
*For ada-002, cos(θ) values range from 0.7-1

💻 Demo: Compare vectors with cosine similarity (aka.ms/aitour/vectors)

🔗 Demo: Vector Embeddings Comparison (aka.ms/aitour/vector-similarity)

# Vector search

1. Compute the embedding vector for the query
2. Find K closest vectors for the query vector
   - Search exhaustively or using approximations




💻 [Demo: Search vectors with query vector](aka.ms/aitour/vectors) ([aka.ms/aitour/vectors](aka.ms/aitour/vectors))

# Vector databases

- Durably store and index vectors and metadata at scale
- Various indexing & retrieval strategies
- Combine vector queries with metadata filters
- Enable access control

PostgreSQL with pgvector example:

```sql
CREATE EXTENSION vector;

CREATE TABLE items (id bigserial PRIMARY KEY,
  embedding vector(1536));

INSERT INTO items (embedding) VALUES
  ('[0.0014701404143124819,
    0.0034404152538627386,
    -0.012805989943444729,...]');

SELECT * FROM items
  ORDER BY
  embedding <=> '[-0.01266181, -0.0279284,...]'
  LIMIT 5;

CREATE INDEX ON items
  USING hnsw (embedding vector_cosine_ops);
```

# Vector databases in Azure

## Vectors in Azure databases

Keep your data where it is:
native vector search capabilities

Built into
Azure Cosmos DB MongoDB vCore and
Azure Cosmos DB for PostgreSQL services

## Azure AI Search

Best relevance: highest quality
of results out of the box

Automatically index data
from Azure data sources:
SQL DB, Cosmos DB, Blob
Storage, ADLSv2, and more

# Azure AI Search

Feature rich, enterprise-ready vector database

Data and platform integration

State-of-the-art retrieval system

*Previously known as Azure Cognitive Search

# Azure AI Search

| Feature-rich vector database | Ingest any data type, from any source | Seamless data & platform integrations | State-of-the-art search ranking | Enterprise-ready foundation |
|---|---|---|---|---|
| Generally available | | Public preview | Generally available | |
| Vector search | | Azure AI Search in Azure AI Studio<br>Integrated vectorization | Semantic ranker | |

# Vector search in Azure AI Search

Feature rich, enterprise-ready

# Vector search in Azure AI Search

Generally available

- Comprehensive vector search solution
- Enterprise-ready

  → scalability, security and compliance
- Integrated with Semantic Kernel, LangChain, LlamaIndex, Azure OpenAI Service, Azure AI Studio, and more

💻 Demo: Azure AI search with vectors (aka.ms/aitour/azure-search)

# Vector search strategies

## ANN search

- ANN = Approximate Nearest Neighbors
- Fast vector search at scale
- Uses HNSW, a graph method with excellent performance-recall profile
- Fine control over index parameters

```
r = search_client.search(
        None,
        top=5,
        vector_queries=[VectorizedQuery(
            vector=search_vector,
            k_nearest_neighbors=5,
            fields="embedding")])
```

## Exhaustive KNN search

- KNN = K Nearest Neighbors
- Per-query or built into schema
- Useful to create recall baselines
- Scenarios with highly selective filters
  - e.g., dense multi-tenant apps

```
r = search_client.search(
        None,
        top=5,
        vector_queries=[VectorizedQuery(
            vector=search_vector,
            k_nearest_neighbors=5,
            fields="embedding",
            exhaustive=True)])
```

# Rich vector search query capabilities

## Filtered vector search

- Scope to date ranges, categories, geographic distances, access control groups, etc.
- Rich filter expressions
- Pre-/post-filtering
  - Pre-filter: great for selective filters, no recall disruption
  - Post-filter: better for low-selectivity filters, but watch for empty results

[https://learn.microsoft.com/azure/search/vector-search-filters](https://learn.microsoft.com/azure/search/vector-search-filters)

```
r = search_client.search(
        None,
        top=5,
        vector_queries=[VectorizedQuery(
            vector=query_vector,
            k_nearest_neighbors=5,
            fields="embedding")],
        vector_filter_mode=VectorFilterMode.PRE_FILTER,
        filter=
    "tag eq 'perks' and created gt 2023-11-15T00:00:00Z")
```

## Multi-vector scenarios

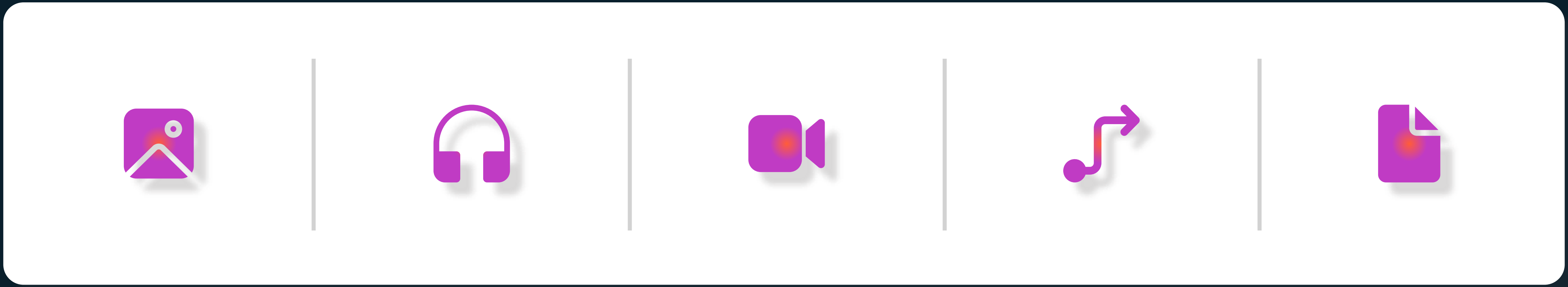
- Multiple vector fields per document
- Multi-vector queries
- Can mix and match as needed

```
r = search_client.search(
        None,
        top=5,
        vector_queries=[
         VectorizedQuery(
             vector=query1, fields=”body_vector",
             k_nearest_neighbors=5,),
         VectorizedQuery(
             vector=query2, fields=”title_vector”,
             k_nearest_neighbors=5,)
        ])
```

# Enterprise ready vector database

| | |
|---|---|
| **Data Encryption** | Including option for customer-managed encryption keys |
| **Secure Authentication** | Managed identity and RBAC support |
| **Network Isolation** | Private endpoints, virtual networks |
| **Compliance Certifications** | Extensive certifications across finance, healthcare, government, etc. |

# Not just text



- Images, sounds, graphs, and more
- Multi-modal embeddings - e.g., images + sentences in Azure AI Vision
- Still vectors → vector search applies
- RAG with images with GPT-4 Turbo with Vision

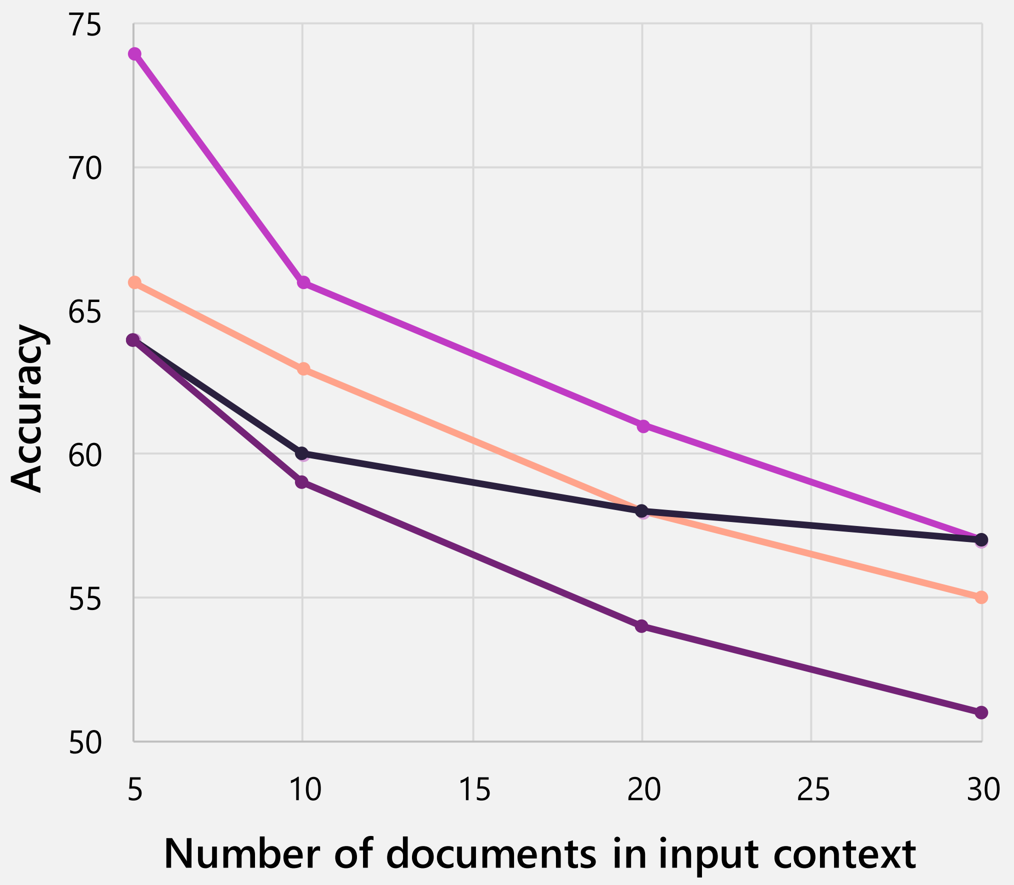
💻 Demo: Searching images (aka.ms/aitour/image-search)

# Azure AI Search:

State-of-the-art retrieval system

# Relevance

- Relevance is critical for RAG apps
- Lots of passages in prompt → degraded quality
  → Can't only focus on recall
- Incorrect passages in prompt → possibly well-grounded yet wrong answers
  → Helps to establish thresholds for "good enough" grounding data




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

# Improving relevance

All information retrieval tricks apply!

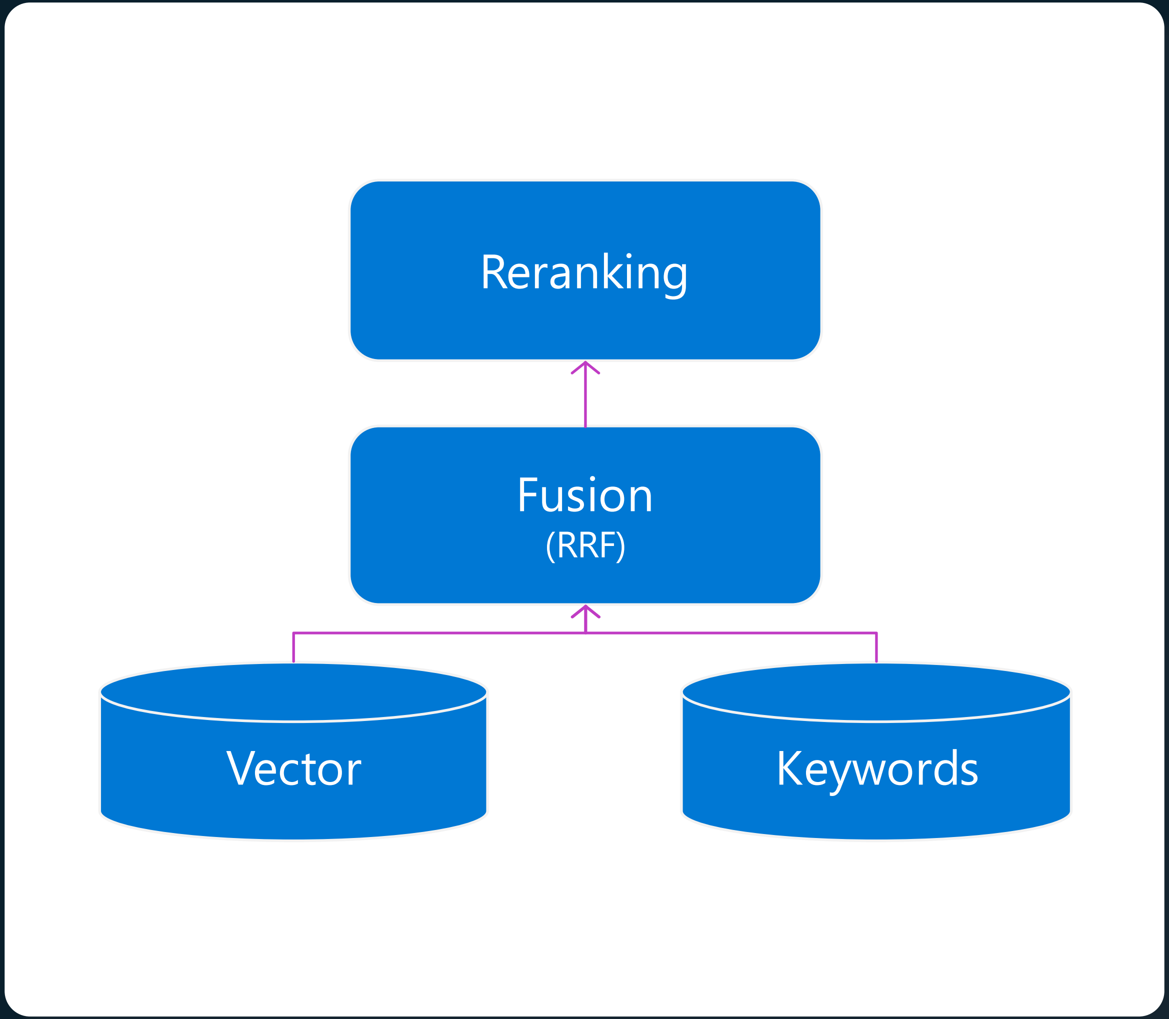
Complete search stacks do better:

- Hybrid retrieval (keywords + vectors) > pure-vector or keyword
- Hybrid + Reranking > Hybrid

Identify good & bad candidates

- Normalized scores from Semantic ranker
- Exclude documents below a threshold

💻 [Demo: Compare text, vector, hybrid, reranker](aka.ms/aitour/search-relevance)
(aka.ms/aitour/search-relevance)



Generally available

# Semantic ranker

SOTA re-ranking model

Highest performing retrieval mode

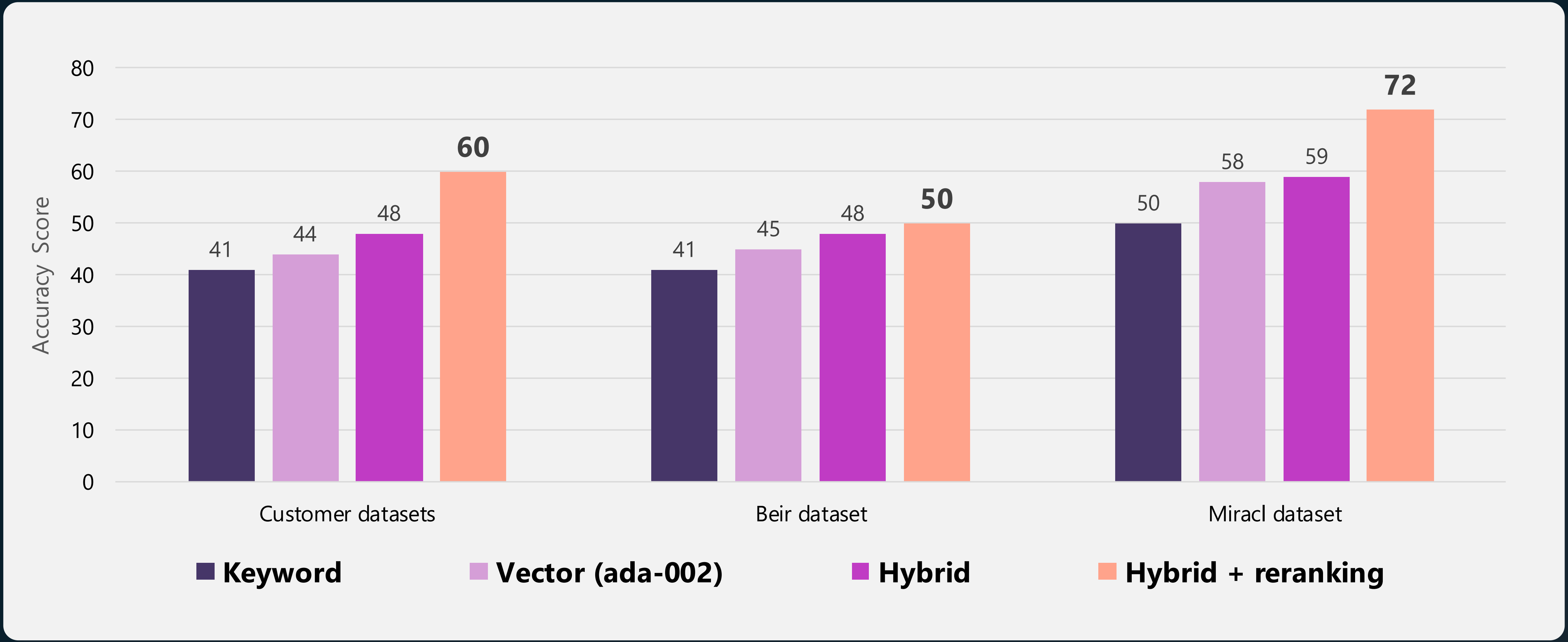
New pay-go pricing: Free 1k requests/month, $1 per additional 1k

Multilingual capabilities

Includes extractive answers, captions and ranking

*Formerly semantic search

# Retrieval relevance across methods




Retrieval comparison using Azure AI Search in various retrieval modes on customer and academic benchmarks
Source: Outperforming vector search with hybrid + reranking

# Impact of query types on relevance

| Query type | Keyword [NDCG@3] | Vector [NDCG@3] | Hybrid [NDCG@3] | Hybrid + Semantic ranker [NDCG@3] |
|---|---|---|---|---|
| Concept seeking queries | 39 | 45.8 | 46.3 | 59.6 |
| Fact seeking queries | 37.8 | 49 | 49.1 | 63.4 |
| Exact snippet search | 51.1 | 41.5 | 51 | 60.8 |
| Web search-like queries | 41.8 | 46.3 | 50 | 58.9 |
| Keyword queries | 79.2 | 11.7 | 61 | 66.9 |
| Low query/doc term overlap | 23 | 36.1 | 35.9 | 49.1 |
| Queries with misspellings | 28.8 | 39.1 | 40.6 | 54.6 |
| Long queries | 42.7 | 41.6 | 48.1 | 59.4 |
| Medium queries | 38.1 | 44.7 | 46.7 | 59.9 |
| Short queries | 53.1 | 38.8 | 53 | 63.9 |

Source: Outperforming vector search with hybrid + reranking

# Azure AI Search:

Seamless Data and Platform Integrations

# Data preparation for RAG applications

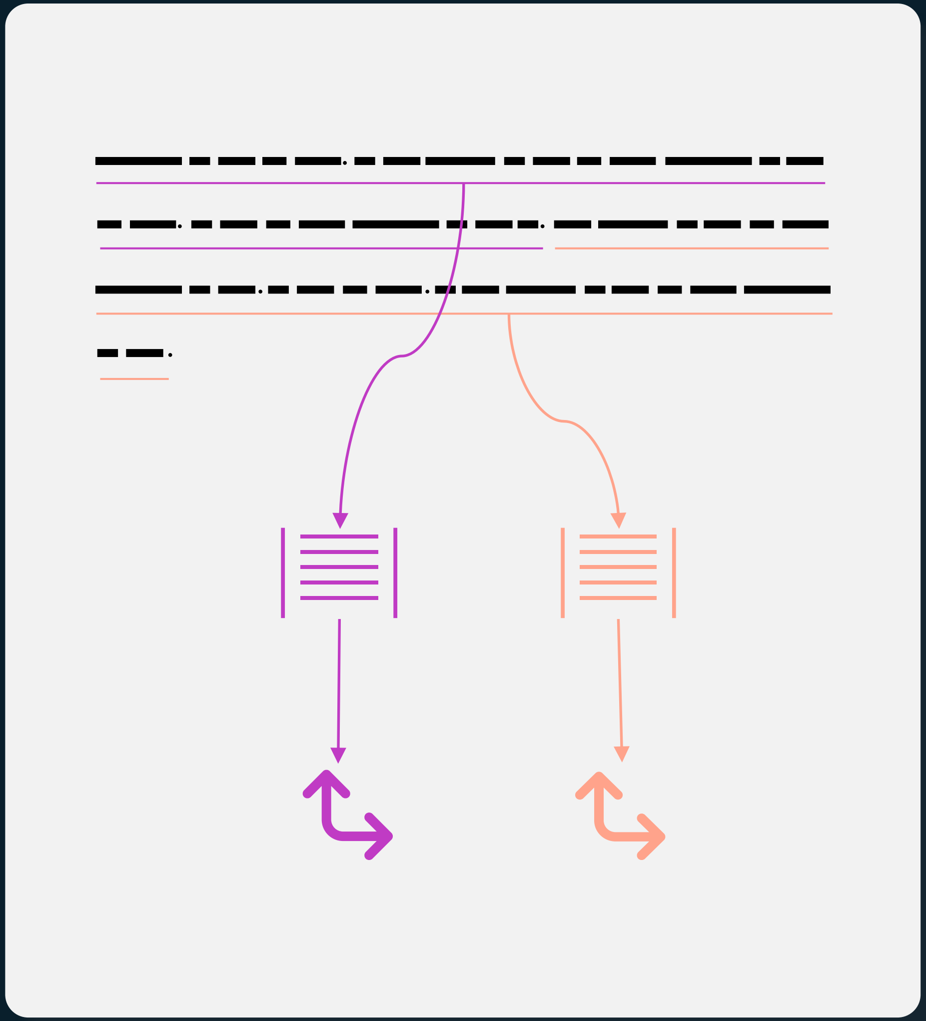
## Chunking

- Split long-form text into short passages
  - LLM context length limits
  - Focused subset of the content
  - Multiple independent passages
- Basics
  - ~200–500 tokens/passage
  - Maintain lexical boundaries
  - Introduce overlap
- Layout
  - Layout information is valuable, e.g., tables

## Vectorization

- Indexing-time: convert passages to vectors

🔗 Example: Data preparation process



# Integrated vectorization

In preview

## End-to-end data processing tailored to RAG

### Data source access

- Blob Storage
- ADLSv2
- SQL DB
- CosmosDB
- …

\+ Incremental change tracking

### File format cracking

- PDFs
- Office documents
- JSON files
- …

\+ Extract images and text, OCR as needed

### Chunking

- Split text into passages
- Propagate document metadata

### Vectorization

- Turn chunks into vectors
- OpenAI embeddings or your custom model

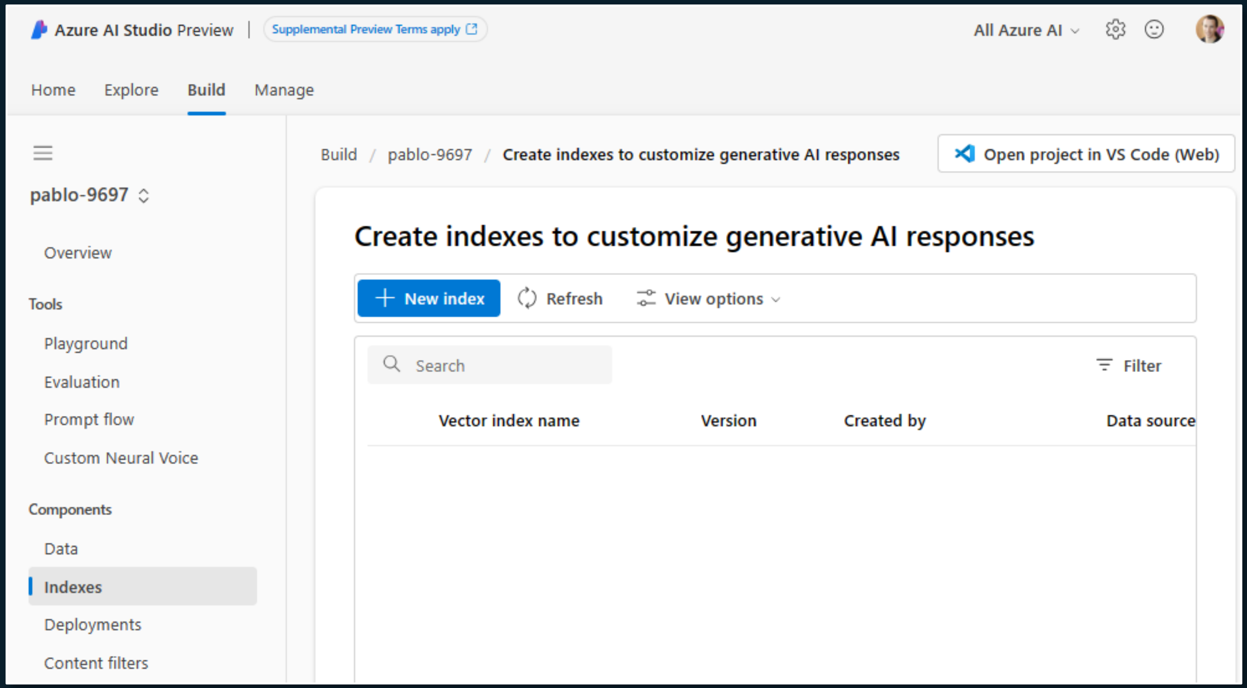
### Indexing

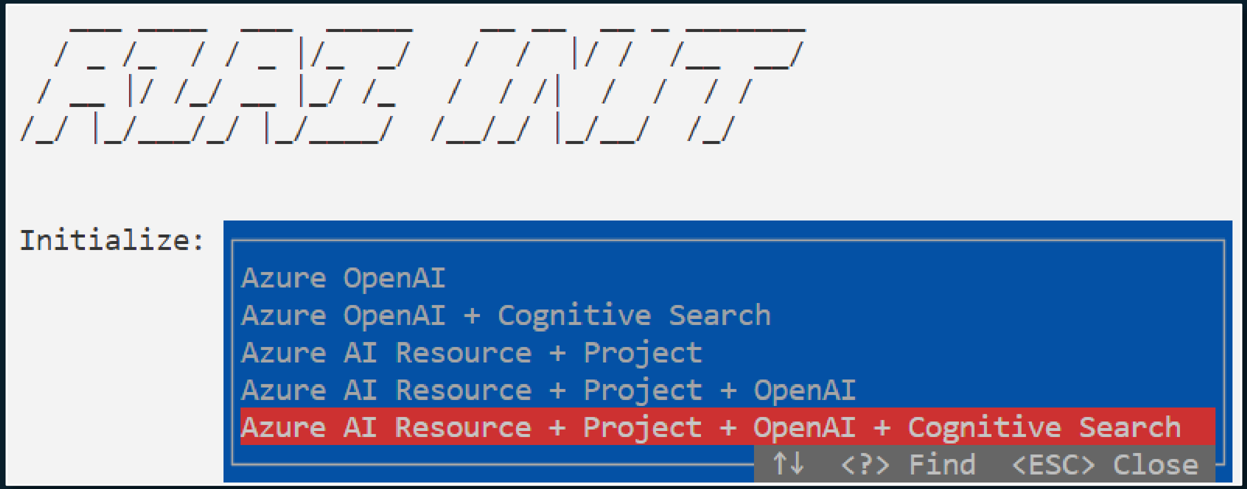
- Document index
- Chunk index
- Both

https://learn.microsoft.com/azure/search/vector-search-integrated-vectorization

# Azure AI Studio & Azure AI SDK

- First-class integration
- Build indexes from data in Blob Storage, Microsoft Fabric, etc.
- Attach to existing Azure AI Search indexes



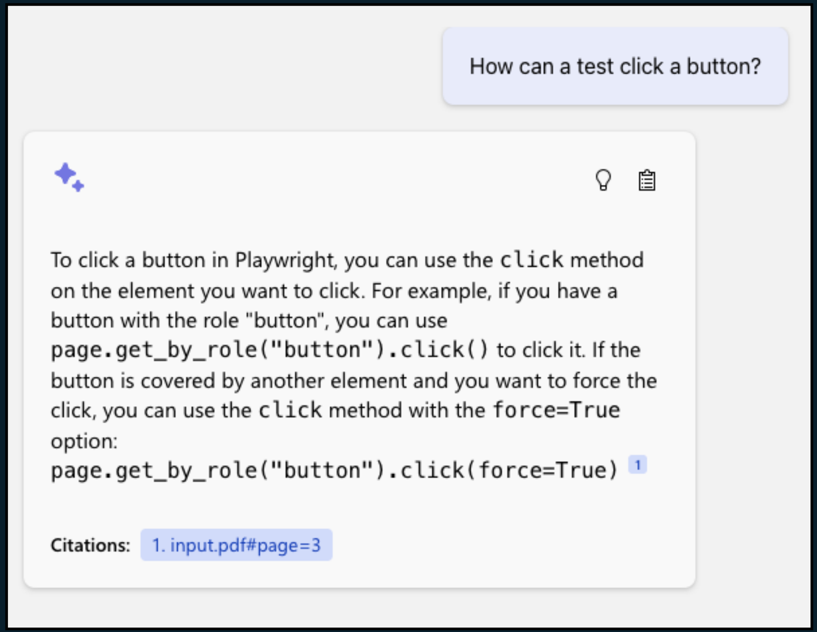



# Use cases

# Example uses

Developers have used Azure AI search to create RAG apps for...

- Public government data
- Internal HR documents, company meetings, presentations
- Customer support requests and call transcripts
- Technical documentation and issue trackers
- Product manuals



# Next steps

Learn more about Azure AI Search

https://aka.ms/AzureAISearch

Dig more into quality evaluation details and why Azure AI Search will make your application generate better results

https://aka.ms/ragrelevance

Deploy a RAG chat application for your organization’s data

https://aka.ms/azai/python

Explore Azure AI Studio for a complete RAG development experience

https://aka.ms/AzureAIStudio

# Join us to learn together!

## Today's workshops:

**Workshop: Developing a production-level RAG workflow**

12:00-1:15pm

2:15-3:30pm

Build a RAG workflow with Prompt Flow, Azure AI Studio, Azure AI Search, Cosmos DB and Azure OpenAI

See you there!

## Upcoming virtual event:


aka.ms/hacktogether/chatapp