Microsoft

RAG Workshop with Azure OpenAI & AI Search

April 2025

Agenda

1 Objectives of the workshop

2 Description of RAG

3 GitHub repository 'rag-workshop' and workshop environment setup

4 Preparation and indexing of database contents

5 Preparation and indexing of file contents

6 Search and response generation

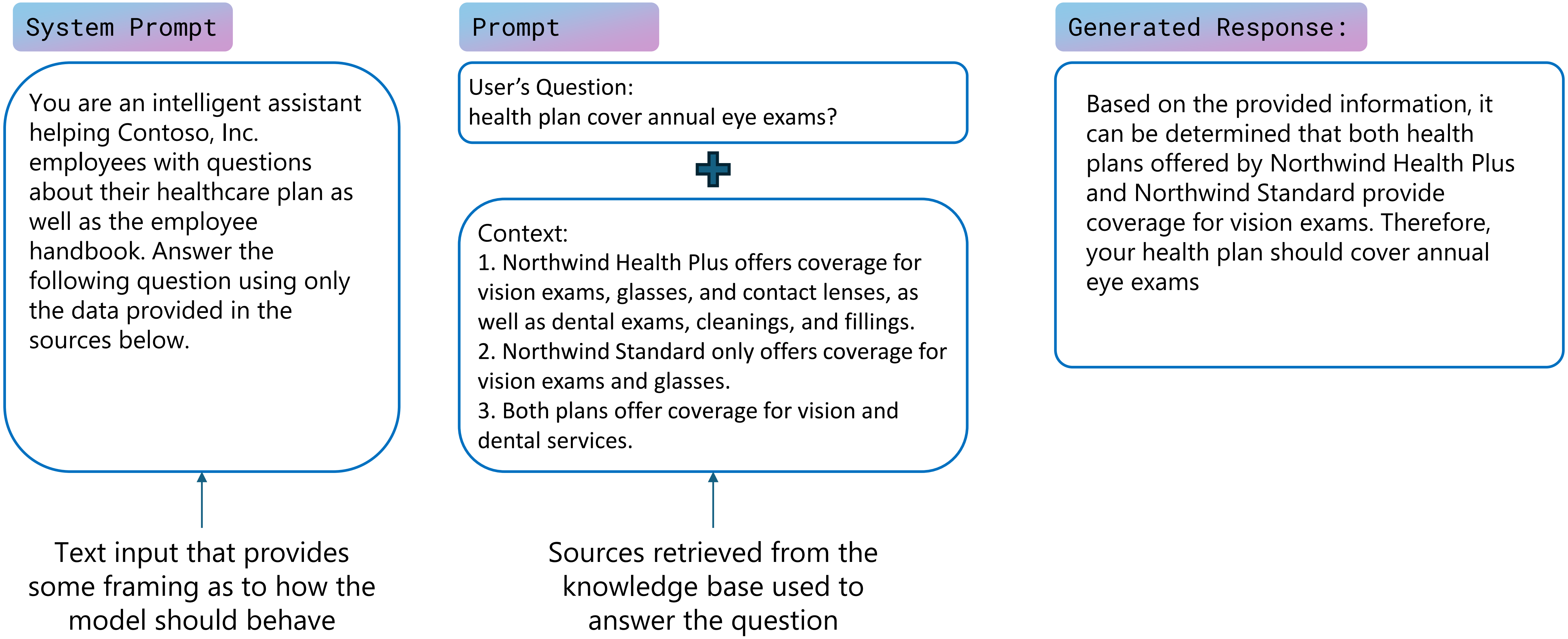
7 Response evaluation with AI Foundry SDK

8 RAG chat demo

1. Objetivos

- Conduct a proof of concept of RAG using actual content from the customer
- Train customer team in the best practices of a RAG Solution
- Formación del equipo de INDITEX en las buenas practicas de una solución RAG
- Agree conclusions and next steps

2. Retrieval Augmented Generation: Bring your data to the prompt



System Prompt

You are an intelligent assistant helping Contoso, Inc. employees with questions about their healthcare plan as well as the employee handbook. Answer the following question using only the data provided in the sources below.

Text input that provides some framing as to how the model should behave

Prompt

User's Question:
health plan cover annual eye exams?

+

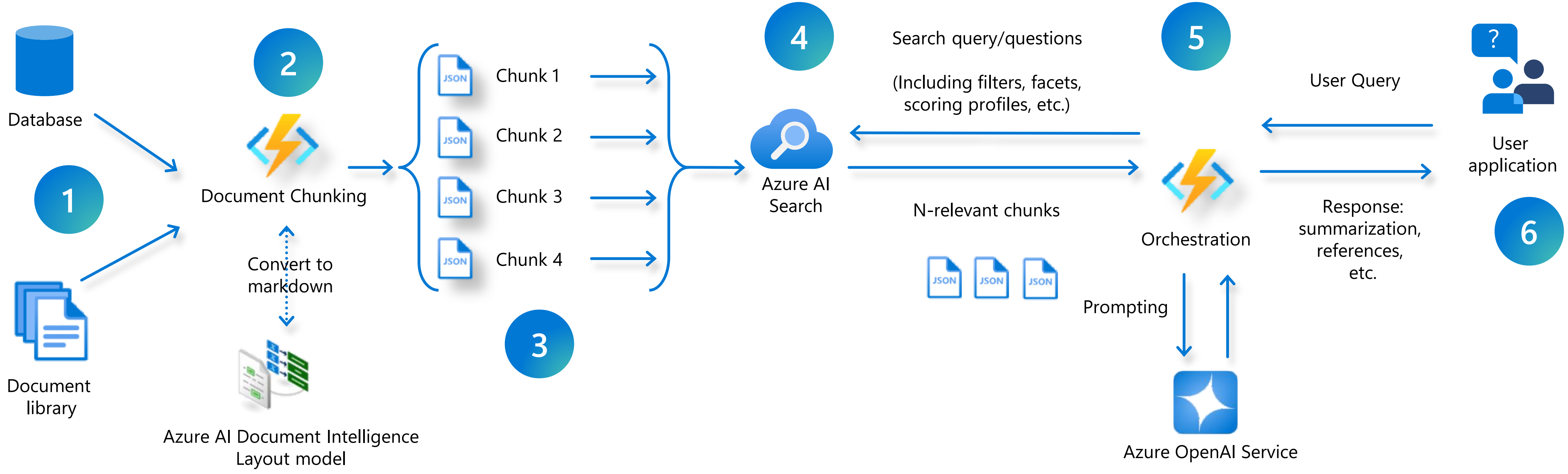
Context:
1. Northwind Health Plus offers coverage for vision exams, glasses, and contact lenses, as well as dental exams, cleanings, and fillings.
2. Northwind Standard only offers coverage for vision exams and glasses.
3. Both plans offer coverage for vision and dental services.

Sources retrieved from the knowledge base used to answer the question

Generated Response:

Based on the provided information, it can be determined that both health plans offered by Northwind Health Plus and Northwind Standard provide coverage for vision exams. Therefore, your health plan should cover annual eye exams

2. Anatomy of RAG



1. Data ingestion

Different data formats and system of records

2. Chunking

Chunking strategy

3. Indexing

Keyword and embedding indexing

4. Data retrieving

Hybrid search with Semantic ranker

5. Augmenting

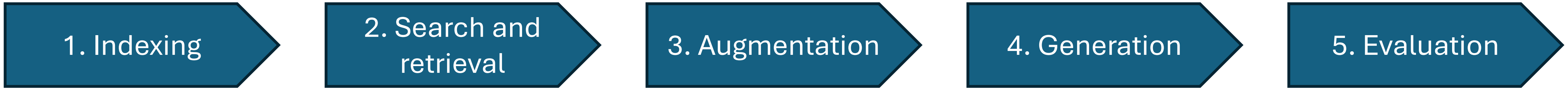
Communication coordination:

- Select most relevant chunks
- Deliver prompt to retriever
- Send response to user app

6. User interface

Chatbot for Q&A surfaced to end users

2. Workflow of RAG



1. Indexing

- Create AI Search indexes
- Get data from a SQLite database
- Get data from an rest API service
- Create chunks and index them

2. Search and retrieval

- Create query with conversation history
- Semantic Hybrid Search

3. Augmentation

- Filter retrieved chunks comparing with user’s question

4. Generation

- Generate answer using the filtered chunks as context and the user’s question

5. Evaluation

- Evaluate the answer comparing with the ground truth

2. Enhancing RAG with Advanced Retrieval Features

Investing in cutting-edge retrieval technology for improved results

R ← The quality of the **retriever** is critical!

A

G

Azure AI Search is committed to providing the BEST retrieval solution through:

- Vector Search capabilities
- Hybrid Search
- Advanced filtering
- Document security
- L2 reranking/optimization
- Built-in chunking
- Auto-Vectorization
- And much more!

Vector Search

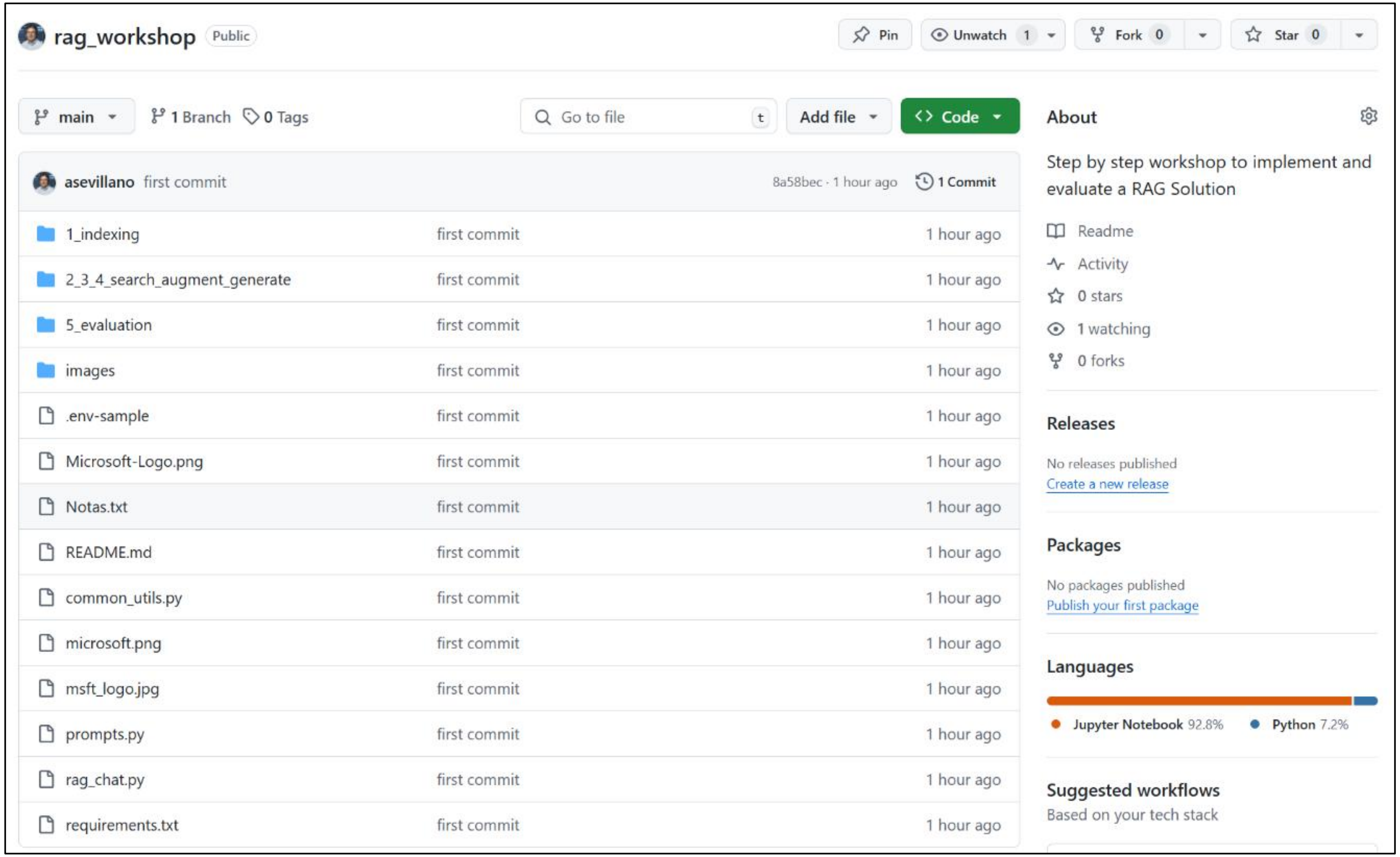
- Exhaustive KNN and ANN search strategies
- Multi-modal, multi-lingual similarity search
- Filters to include or exclude information
- End-to-end data ingestion, chunking, vectorization and retrieval
- Integrated with
  Semantic kernel, LangChain, Azure OpenAI Service and AzureML Promptflow

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 (like Bing)

Github repo ‘rag-workshop’

https://github.com/asevillano/rag_workshop



asevillano first commit 8a58bec · 1 hour ago 1 Commit

1_indexing	first commit	1 hour ago
2_3_4_search_augment_generate	first commit	1 hour ago
5_evaluation	first commit	1 hour ago
images	first commit	1 hour ago
.env-sample	first commit	1 hour ago
Microsoft-Logo.png	first commit	1 hour ago
Notas.txt	first commit	1 hour ago
README.md	first commit	1 hour ago
common_utils.py	first commit	1 hour ago
microsoft.png	first commit	1 hour ago
msft_logo.jpg	first commit	1 hour ago
prompts.py	first commit	1 hour ago
rag_chat.py	first commit	1 hour ago
requirements.txt	first commit	1 hour ago

About

Step by step workshop to implement and evaluate a RAG Solution

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● Jupyter Notebook 92.8% ● Python 7.2%

Suggested workflows
Based on your tech stack

Preparation of Azure Machine Learning’s notebook

1. Create the Azure Machine Learning service
2. Open Azure ML Studio
3. Create the .env file using .env-template with the configuration parameters
4. Upload code: Notebooks > Files > (+) Add Files > Upload folder > rag_workshop
5. Create compute (todo por defecto)
6. Elegir ‘compute’ > servidor y Python 3.10 SDK 2
7. Install python packages with ‘%pip install …’ in indexing.ipynb

Creation of Azure services

1. Create Azure OpenAI service:
   - Get end-point and api key, and edit `AZURE_OPENAI_ENDPOINT` and `AZURE_OPENAI_API_KEY` variables in .env file
2. Open Azure AI Foundry and deploy needed models:
   - text-embedding-ada-002 to calculate vectors, edit `AZURE_OPENAI_EMBEDDING_DEPLOYMENT_NAME` variable
   - gpt-4o-mini to evaluate chunks against the user question, edit `AZURE_OPENAI_RERANK_DEPLOYMENT_NAME` variable
   - gpt-4o to generate answers, edit `AZURE_OPENAI_DEPLOYMENT_NAME` variable

1. Create Azure AI Search service:
   - Get end-point and api key, variables `SEARCH_SERVICE_ENDPOINT` and `SEARCH_SERVICE_QUERY_KEY`
   - Set index names for database and document contents in `SEARCH_INDEX_NAME_REGS` and `SEARCH_INDEX_NAME_DOCS`

1. Create Document Intelligence service:
   - Get end-point and api key, and edit `DOC_INTEL_ENDPOINT` y `DOC_INTEL_KEY` variables in .env file

1. Configure PostgreSQL connection:
   - Edit variables in.env file: `PG_HOST`, `PG_PORT`, `PG_USER`, `PG_PASSWORD`, `PG_DATABASE`

1. Upload .env file in the Azure ML notebook

3. Prepare and index database contents (I)

Notebook: 1_indexing/indexing.ipynb

Example of getting data from a PostgreSQL database:

- `query_pg`: function to get data from a PostgreSQL database executing a SQL query.
- Adapt connection variables in .env file:

Functions for indexing and searching:

- `create_index`: create AI Search index with title and content text fields, and embeddingTitle and embeddingContent vector fields.
- `chunk_text`: split (chunk) content with fix size of 512 tokens with 25% of overlapping
- `index_documents`: index chunks in AI Search
- `semantic_hybrid_search`: hybrid with semantic ranking search test

3. Prepare and index database contents (II)

Notebook: 1_indexing/indexing.ipynb

Example of getting data from a database thru an end-point:

- `query_sqlite_endpoint`: adaptar la función para obtener datos de una base de datos ejecutando una query SQL con una petición a un end-point API REST.

Functions for indexing and searching:

- `create_index`: create AI Search index with title and content text fields, and embeddingTitle and embeddingContent vector fields.
- `chunk_text`: split (chunk) content with fix size of 512 tokens with 25% of overlapping
- `index_documents`: index chunks in AI Search
- `semantic_hybrid_search`: hybrid with semantic ranking search test

4. Prepare and index file content

Notebook: 1_indexing/indexing.ipynb

- `create_index`: create AI Search index with title and content text fields, and embeddingTitle and embeddingContent vector fields.
- `process_files`: convert PDFs files to markdown format
- `chunk_and_index_md_files`: split (chunk) markdown files with fix size of 512 tokens with 25% of overlapping, and chunk indexing in AI Search
- `semantic_hybrid_search`: hybrid with semantic ranking search test

5. Search, Augment and Answer Generation

Notebook: 2_3_4_search_augment_generate/search_augment_generate.ipynb

- `generate_search_query`: prepare the search query with conversation history.
- `semantic_hybrid_search`: hybrid with semantic ranking search.
- `get_filtered_chunks`: filtering of less relevant chunks for the user’s query.
- `generate_answer_with_history`: generate of answer with Azure OpenAI using the most relevant chunks and conversation history.

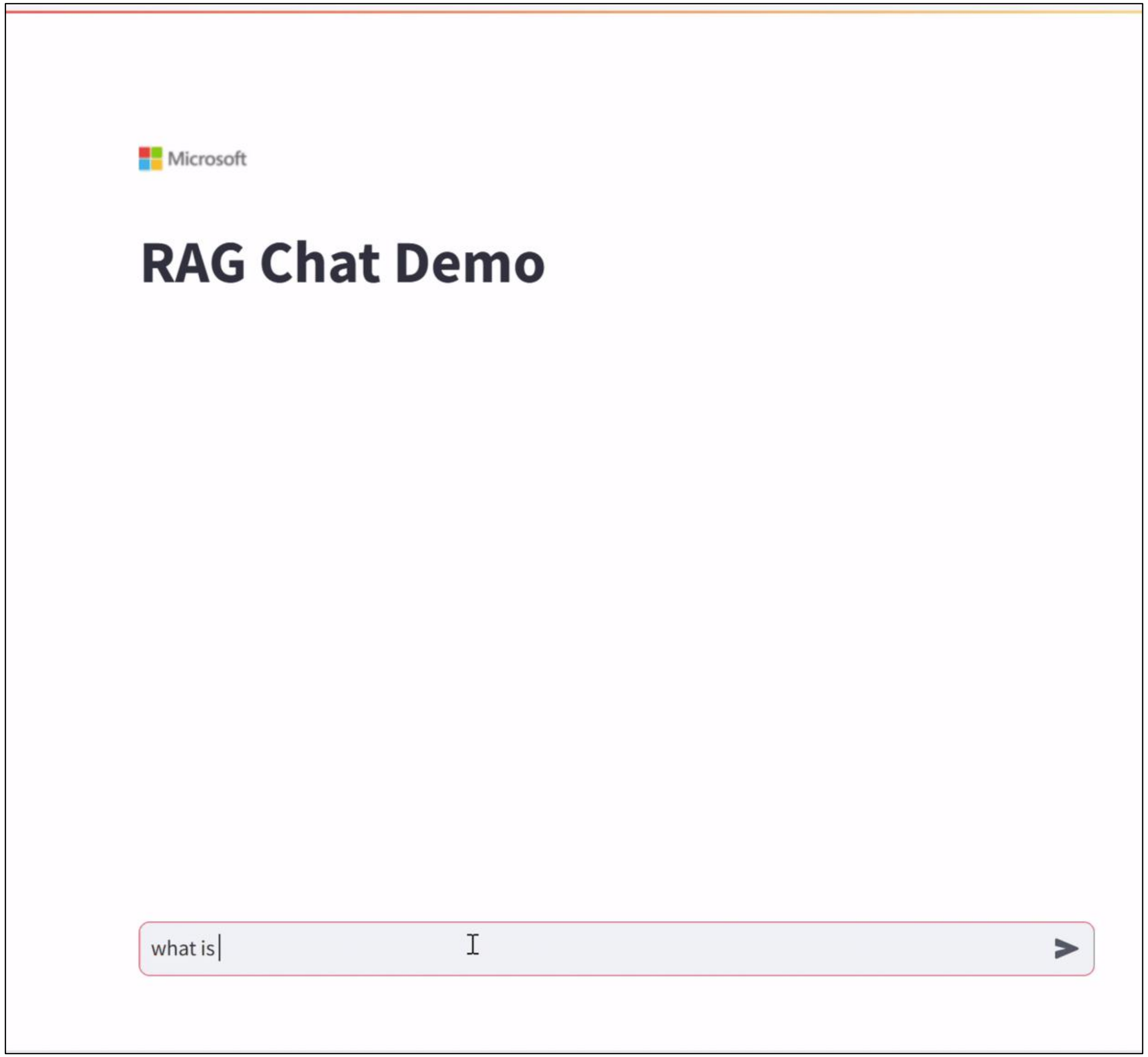
6. Evaluate answers with AI Foundry’s SDK

Notebook: 5_evaluation/ evaluation.ipynb

- `generate_search_query`: prepare the search query with conversation history.
- `semantic_hybrid_search`: hybrid with semantic ranking search.
- `get_filtered_chunks`: filtering of less relevant chunks for the user’s query.
- `generate_answer_with_history`: generate of answer with Azure OpenAI using the most relevant chunks and conversation history.
- `evaluate_answer`: evaluate answers compared with the ground truth in an Excel file, with the AI Foundry’s SDK metrics

7. Demo RAG chat

To execute the demo chat, run the following command: `streamlit run rag_chat.py`



Microsoft

RAG Chat Demo

what is