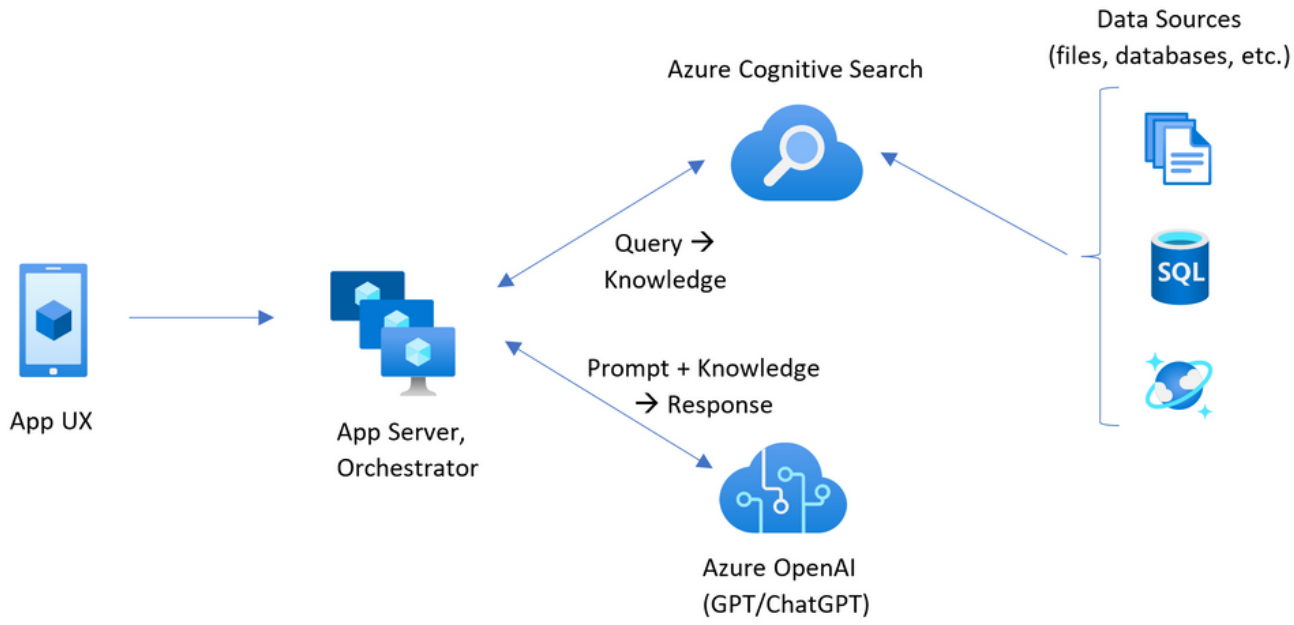
ChatGPT + Enterprise data with Azure OpenAI and Cognitive Search

GITHUB CODESPACES OPEN REMOTE - CONTAINERS OPEN

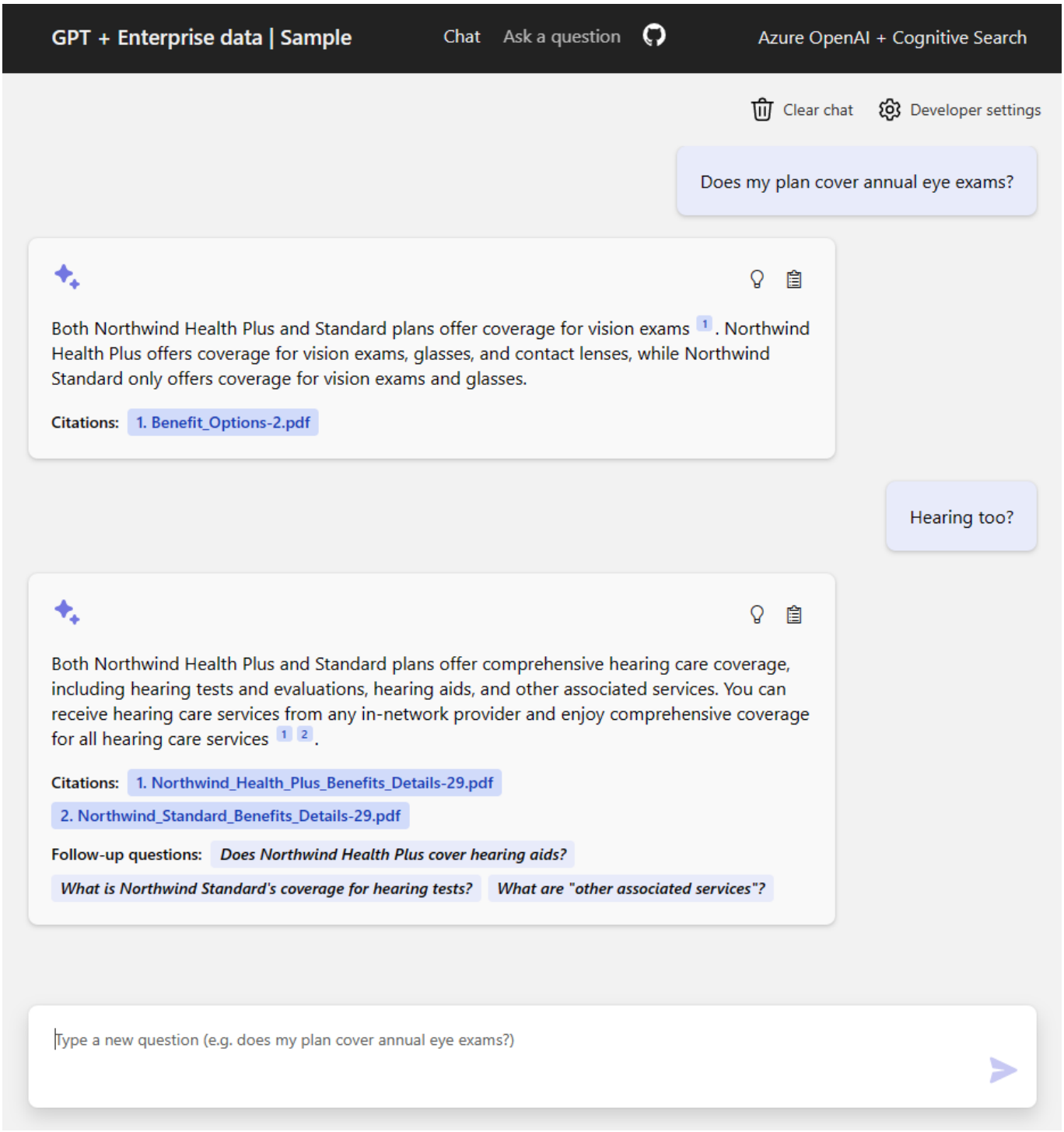
This sample demonstrates a few approaches for creating ChatGPT-like experiences over your own data using the Retrieval Augmented Generation pattern. It uses Azure OpenAI Service to access the ChatGPT model (gpt-35-turbo), and Azure Cognitive Search for data indexing and retrieval.

The repo includes sample data so it's ready to try end to end. In this sample application we use a fictitious company called Contoso Electronics, and the experience allows its employees to ask questions about the benefits, internal policies, as well as job descriptions and roles.



Features

- Chat and Q&A interfaces
- Explores various options to help users evaluate the trustworthiness of responses with citations, tracking of source content, etc.
- Shows possible approaches for data preparation, prompt construction, and orchestration of interaction between model (ChatGPT) and retriever (Cognitive Search)
- Settings directly in the UX to tweak the behavior and experiment with options



Getting Started

> **IMPORTANT:** In order to deploy and run this example, you'll need an **Azure subscription with access enabled for the Azure OpenAI service**. You can request access here. You can also visit here to get some free Azure credits to get you started.

> **AZURE RESOURCE COSTS** by default this sample will create Azure App Service and Azure Cognitive Search resources that have a monthly cost, as well as Form Recognizer resource that has cost per document page. You can switch them to free versions of each of them if you want to avoid this cost by changing the parameters file under the infra folder (though there are some limits to consider; for example, you can have up to 1 free Cognitive Search resource per subscription, and the free Form Recognizer resource only analyzes the first 2 pages of each document.)

Prerequisites

To Run Locally

- Azure Developer CLI
- Python 3+
  - **Important**: Python and the pip package manager must be in the path in Windows for the setup scripts to work.
  - **Important**: Ensure you can run `python --version` from console. On Ubuntu, you might need to run `sudo apt install python-is-python3` to link `python` to `python3`.
- Node.js
- Git
- Powershell 7+ (pwsh) - For Windows users only.
  - **Important**: Ensure you can run `pwsh.exe` from a PowerShell command. If this fails, you likely need to upgrade PowerShell.

> NOTE: Your Azure Account must have `Microsoft.Authorization/roleAssignments/write` permissions, such as User Access Administrator or Owner.

To Run in GitHub Codespaces or VS Code Remote Containers

You can run this repo virtually by using GitHub Codespaces or VS Code Remote Containers. Click on one of the buttons below to open this repo in one of those options.

Installation

Project Initialization

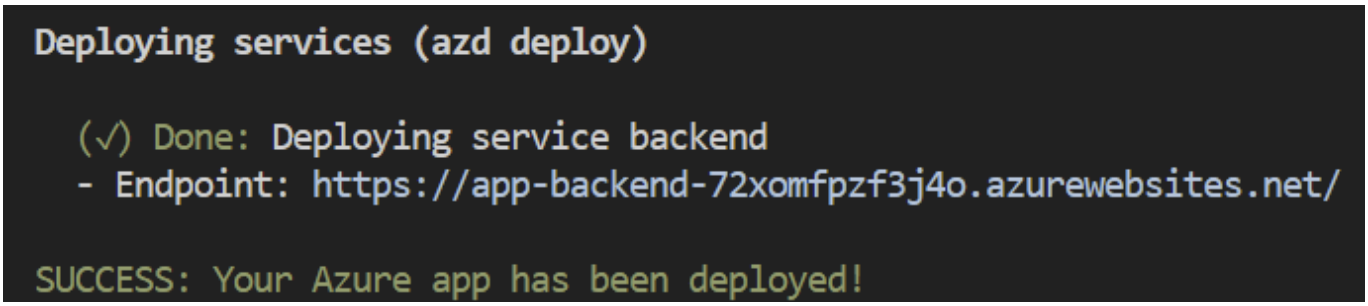
1. Create a new folder and switch to it in the terminal
2. Run `azd auth login`
3. Run `azd init -t azure-search-openai-demo`
   - For the target location, the regions that currently support the models used in this sample are **East US** or **South Central US**. For an up-to-date list of regions and models, check here
   - note that this command will initialize a git repository and you do not need to clone this repository

Starting from scratch:

Execute the following command, if you don't have any pre-existing Azure services and want to start from a fresh deployment.

1. Run `azd up` - This will provision Azure resources and deploy this sample to those resources, including building the search index based on the files found in the `./data` folder.
2. After the application has been successfully deployed you will see a URL printed to the console. Click that URL to interact with the application in your browser.

It will look like the following:



> NOTE: It may take a minute for the application to be fully deployed. If you see a "Python Developer" welcome screen, then wait a minute and refresh the page.

Use existing resources:

1. Run `azd env set AZURE_OPENAI_SERVICE {Name of existing OpenAI service}`
2. Run `azd env set AZURE_OPENAI_RESOURCE_GROUP {Name of existing resource group that OpenAI service is provisioned to}`
3. Run `azd env set AZURE_OPENAI_CHATGPT_DEPLOYMENT {Name of existing ChatGPT deployment}`. Only needed if your ChatGPT deployment is not the default 'chat'.
4. Run `azd env set AZURE_OPENAI_GPT_DEPLOYMENT {Name of existing GPT deployment}`. Only needed if your ChatGPT deployment is not the default 'davinci'.
5. Run `azd up`

> NOTE: You can also use existing Search and Storage Accounts. See `./infra/main.parameters.json` for list of environment variables to pass to `azd env set` to configure those existing resources.

Deploying or re-deploying a local clone of the repo:

- Simply run `azd up`

Running locally:

1. Run `azd login`
2. Change dir to `app`
3. Run `./start.ps1` or `./start.sh` or run the "VS Code Task: Start App" to start the project locally.

Sharing Environments

Run the following if you want to give someone else access to completely deployed and existing environment.

1. Install the Azure CLI
2. Run `azd init -t azure-search-openai-demo`
3. Run `azd env refresh -e {environment name}` - Note that they will need the azd environment name, subscription Id, and location to run this command - you can find those values in your `./azure/{env name}/.env` file. This will populate their azd environment's .env file with all the settings needed to run the app locally.
4. Run `pwsh ./scripts/roles.ps1` - This will assign all of the necessary roles to the user so they can run the app locally. If they do not have the necessary permission to create roles in the subscription, then you may need to run this script for them. Just be sure to set the `AZURE_PRINCIPAL_ID`

environment variable in the azd .env file or in the active shell to their Azure Id, which they can get with `az account show`.

Quickstart

- In Azure: navigate to the Azure WebApp deployed by azd. The URL is printed out when azd completes (as "Endpoint"), or you can find it in the Azure portal.
- Running locally: navigate to 127.0.0.1:5000

Once in the web app:

- Try different topics in chat or Q&A context. For chat, try follow up questions, clarifications, ask to simplify or elaborate on answer, etc.
- Explore citations and sources
- Click on "settings" to try different options, tweak prompts, etc.

Resources

- Revolutionize your Enterprise Data with ChatGPT: Next-gen Apps w/ Azure OpenAI and Cognitive Search
- Azure Cognitive Search
- Azure OpenAI Service

Note

> Note: The PDF documents used in this demo contain information generated using a language model (Azure OpenAI Service). The information contained in these documents is only for demonstration purposes and does not reflect the opinions or beliefs of Microsoft. Microsoft makes no representations or warranties of any kind, express or implied, about the completeness, accuracy, reliability, suitability or availability with respect to the information contained in this document. All rights reserved to Microsoft.

FAQ

Question: Why do we need to break up the PDFs into chunks when Azure Cognitive Search supports searching large documents?

Answer: Chunking allows us to limit the amount of information we send to OpenAI due to token limits. By breaking up the content, it allows us to easily find potential chunks of text that we can inject into OpenAI. The method of chunking we use leverages a sliding window of text such that sentences that end one chunk will start the next. This allows us to reduce the chance of losing the context of the text.

Troubleshooting

If you see this error while running `azd deploy`: `read /tmp/azd1992237260/backend_env/lib64: is a directory`, then delete the `./app/backend/backend_env folder` and re-run the `azd deploy` command. This issue is being tracked here: https://github.com/Azure/azure-dev/issues/1237

If the web app fails to deploy and you receive a '404 Not Found' message in your browser, run 'azd deploy'.