Purpose of the Repo:

The `azure-search-openai-demo` repository demonstrates how to integrate Azure Cognitive Search with OpenAI's GPT-3 to create an intelligent search application. The goal is to enhance search capabilities by combining the robust document indexing and retrieval features of Azure Cognitive Search with the natural language understanding and generation capabilities of OpenAI. The application uses a Flask web server to provide a user interface for searching and retrieving information, offering a powerful tool for querying and summarizing large datasets.

Azure Cognitive Search:

Azure Cognitive Search is a cloud-based search service provided by Microsoft Azure. It enables developers to build sophisticated search experiences into web and mobile applications. Key features include:

- Full-Text Search: Advanced text search capabilities with support for multiple languages.
- AI-Enriched Indexing: Integration with cognitive skills like language detection, entity recognition, and sentiment analysis.
- Scalability: Automatically scales to handle large datasets and high query loads.
- Customizable: Allows for the customization of search indexes and ranking models.

Azure Cognitive Search enhances the searchability of large datasets and provides enriched search results using AI-driven techniques.

Readme of the repo:

1. Overview

The repository demonstrates an integration of Azure Cognitive Search with OpenAI, showcasing how to build an intelligent search application. It combines the robust search capabilities of Azure Cognitive Search with the natural language understanding of OpenAI's models.

2. Prerequisites

- **Azure Subscription**: Required to create resources.
- **OpenAI API Key**: Needed for accessing OpenAI's GPT models.
- **Python**: Install Python 3.7 or later.

3. Setup

Clone the Repository:

```
git clone
https://github.com/Azure-Samples/azure-search-openai-demo.git
cd azure-search-openai-demo
```

Install Dependencies:

```
pip install -r requirements.txt
```

4. Configuration

- **Create and Configure Azure Resources**: Set up Azure Cognitive Search and OpenAI services.
- **Update Configuration**: Rename `config.json.example` to `config.json` and update it with your Azure and OpenAI credentials.

5. Index Data

Run Indexing Script:

```
python app/backend/index_data.py
```

- This script indexes sample data into Azure Cognitive Search.

6. Run the Application

Start Flask Application:
`python app/backend/app.py`

- This will start the web server, typically accessible at `http://localhost:5000`.

7. Usage

- **Search Interface**: Use the web interface to input queries. The application retrieves documents using Azure Cognitive Search and processes them with OpenAI to generate responses.

8. Approaches

- The repository includes different approaches to integrate search and language processing:
    - **chatreadretrieveread**: Chat-based approach for retrieving and reading documents.
    - **readdecomposeask**: Decomposes the query into sub-queries.
    - **readretrieveask**: Retrieves documents and answers queries directly.

Summary

The repository provides a hands-on example of how to combine Azure Cognitive Search with OpenAI to build advanced search applications, leveraging both search indexing and AI-based query processing.

OpenAI API Key Step:

To get and configure an OpenAI API key for the `azure-search-openai-demo`:

Steps to Obtain an OpenAI API Key

1. **Sign Up / Log In:**
    - **Go to the OpenAI website.**
    - **Sign up for a new account or log in to your existing account.**
2. **Navigate to API Keys:**
    - **Once logged in, go to the API Keys section.**
    - **Click on "Create new secret key" to generate a new API key.**
3. **Copy the API Key:**
    - **Copy the generated API key. This key will be used in your application to authenticate requests to OpenAI.**

Configure the Application

1. **Update `config.json`:**
    - **Rename `config.json.example` to `config.json` if not already done.**

Add your OpenAI API key to the configuration file:
json
Copy code

```
{

  "OPENAI_API_KEY": "your-api-key-here",

  "AZURE_SEARCH_SERVICE_NAME": "your-search-service-name",

  "AZURE_SEARCH_INDEX_NAME": "your-search-index-name",

  "AZURE_SEARCH_API_KEY": "your-azure-search-api-key"

}
```

    -

2. **Save the File:**
    - **Ensure the `config.json` file is saved in the `app/backend` directory.**

Summary

1. **Obtain API Key: From OpenAI's API Keys section.**
2. **Update Configuration: Add the key to the `config.json` file.**
3. **Run Application: With the key configured, you can now run the Flask application which will use the OpenAI API for processing queries.**

This ensures your application can authenticate and interact with OpenAI's GPT models for enhanced search functionality.