Create a Question Answering Solution

One of the most common conversational scenarios is providing support through a knowledge base of frequently asked questions (FAQs). Many organizations publish FAQs as documents or web pages, which works well for a small set of question and answer pairs, but large documents can be difficult and time-consuming to search.

Azure AI Language includes a *question answering* capability that enables you to create a knowledge base of question and answer pairs that can be queried using natural language input, and is most commonly used as a resource that a bot can use to look up answers to questions submitted by users.

Provision an *Azure AI Language* resource

If you don't already have one in your subscription, you'll need to provision an **Azure AI Language service** resource. Additionally, to create and host a knowledge base for question answering, you need to enable the **Question Answering** feature.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. In the search field at the top enter **Azure AI services**, then press **Enter**.
3. Select **Create** under the **Language Service** resource in the results.
4. **Select** the **Custom question answering** block. Then select **Continue to create your resource**. You will need to enter the following settings:
    - **Subscription**: *Your Azure subscription*
    - **Resource group**: *Choose or create a resource group*.
    - **Region**: *Choose any available location*
    - **Name**: *Enter a unique name*
    - **Pricing tier**: Select **F0** (*free*), or **S** (*standard*) if F is not available.
    - **Azure Search region**: *Choose a location in the same global region as your Language resource*
    - **Azure Search pricing tier**: Free (F) (*If this tier is not available, select Basic (B)*)
    - **Responsible AI Notice**: *Agree*
5. Select **Create + review**, then select **Create**.

    NOTE Custom Question Answering uses Azure Search to index and query the knowledge base of questions and answers.

6. Wait for deployment to complete, and then go to the deployed resource.
7. View the **Keys and Endpoint** page. You will need the information on this page later in the exercise.

Create a question answering project

To create a knowledge base for question answering in your Azure AI Language resource, you can use the Language Studio portal to create a question answering project. In this case, you'll create a knowledge base containing questions and answers about Microsoft Learn.

1. In a new browser tab, go to the Language Studio portal at https://language.cognitive.azure.com/ and sign in using the Microsoft account associated with your Azure subscription.
2. If you're prompted to choose a Language resource, select the following settings:
    - **Azure Directory**: The Azure directory containing your subscription.
    - **Azure subscription**: Your Azure subscription.
    - **Resource type**: Language
    - **Resource name**: The Azure AI Language resource you created previously.

    If you are not prompted to choose a language resource, it may be because you have multiple Language resources in your subscription; in which case:

e. On the bar at the top if the page, select the **Settings (⚙)** button.
    f. On the **Settings** page, view the **Resources** tab.
    g. Select the language resource you just created, and click **Switch resource**.
    h. At the top of the page, click **Language Studio** to return to the Language Studio home page.
3. At the top of the portal, in the **Create new** menu, select **Custom question answering**.
4. In the ***Create a project** wizard, on the **Choose language setting** page, select the option to **Set the language for all projects in this resource**, and select **English** as the language. Then select **Next**.
5. On the **Enter basic information** page, enter the following details:
    - **Name** `LearnFAQ`
    - **Description**: `FAQ for Microsoft Learn`
    - **Default answer when no answer is returned**: `Sorry, I don't understand the question`
6. Select **Next**.
7. On the **Review and finish** page, select **Create project**.

Add sources to the knowledge base

You can create a knowledge base from scratch, but it's common to start by importing questions and answers from an existing FAQ page or document. In this case, you'll import data from an existing FAQ web page for Microsoft learn, and you'll also import some pre-defined "chit chat" questions and answers to support common conversational exchanges.

1. On the **Manage sources** page for your question answering project, in the ➕ **Add source** list, select **URLs**. Then in the **Add URLs** dialog box, select ➕ **Add url** and set the following name and URL before you select **Add all** to add it to the knowledge base:
    - **Name**: `Learn FAQ Page`
    - **URL**: `https://docs.microsoft.com/en-us/learn/support/faq`
2. On the **Manage sources** page for your question answering project, in the ➕ **Add source** list, select **Chitchat**. The in the **Add chit chat** dialog box, select **Friendly** and select **Add chit chat**.

Edit the knowledge base

Your knowledge base has been populated with question and answer pairs from the Microsoft Learn FAQ, supplemented with a set of conversational *chit-chat* question and answer pairs. You can extend the knowledge base by adding additional question and answer pairs.

1. In your **LearnFAQ** project in Language Studio, select the **Edit knowledge base** page to see the existing question and answer pairs (if some tips are displayed, read them and choose **Got it** to dismiss them, or select **Skip all**)
2. In the knowledge base, on the **Question answer pairs** tab, select ➕, and create a new question answer pair with the following settings:
    - **Source**: `https://docs.microsoft.com/en-us/learn/support/faq`
    - **Question**: `What are Microsoft credentials?`
    - **Answer**: `Microsoft credentials enable you to validate and prove your skills with Microsoft technologies.`
3. Select **Done**.
4. In the page for the **What are Microsoft credentials?** question that is created, expand **Alternate questions**. Then add the alternate question `How can I demonstrate my Microsoft technology skills?`.

    In some cases, it makes sense to enable the user to follow up on an answer by creating a *multi-turn* conversation that enables the user to iteratively refine the question to get to the answer they need.

5. Under the answer you entered for the certification question, expand **Follow-up prompts** and add the following follow-up prompt:

    - **Text displayed in the prompt to the user**: `Learn more about credentials`.
    - Select the **Create link to new pair** tab, and enter this text: `You can learn more about credentials on the [Microsoft credentials page](https://docs.microsoft.com/learn/credentials/).`
    - Select **Show in contextual flow only**. This option ensures that the answer is only ever returned in the context of a follow-up question from the original certification question.
6. Select **Add prompt**.

Train and test the knowledge base

Now that you have a knowledge base, you can test it in Language Studio.

1. Save the changes to your knowledge base by selecting the **Save** button under the **Question answer pairs** tab on the left.
2. After the changes have been saved, select the **Test** button to open the test pane.
3. In the test pane, at the top, deselect **Include short answer response** (if not already unselected). Then at the bottom enter the message `Hello`. A suitable response should be returned.
4. In the test pane, at the bottom enter the message `What is Microsoft Learn?`. An appropriate response from the FAQ should be returned.
5. Enter the message `Thanks!` An appropriate chit-chat response should be returned.
6. Enter the message `Tell me about Microsoft credentials`. The answer you created should be returned along with a follow-up prompt link.
7. Select the **Learn more about credentials** follow-up link. The follow-up answer with a link to the certification page should be returned.
8. When you're done testing the knowledge base, close the test pane.

Deploy the knowledge base

The knowledge base provides a back-end service that client applications can use to answer questions. Now you are ready to publish your knowledge base and access its REST interface from a client.

1. In the **LearnFAQ** project in Language Studio, select the **Deploy knowledge base** page.
2. At the top of the page, select **Deploy**. Then select **Deploy** to confirm you want to deploy the knowledge base.

3. When deployment is complete, select **Get prediction URL** to view the REST endpoint for your knowledge base and note that the sample request includes parameters for:
    - **projectName**: The name of your project (which should be *LearnFAQ*)
    - **deploymentName**: The name of your deployment (which should be *production*)
4. Close the prediction URL dialog box.

Prepare to develop an app in Visual Studio Code

You'll develop your question answering app using Visual Studio Code. The code files for your app have been provided in a GitHub repo.

Tip: If you have already cloned the **mslearn-ai-language** repo, open it in Visual Studio code. Otherwise, follow these steps to clone it to your development environment.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/mslearn-ai-language` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo.

    Note: If you are prompted to add required assets to build and debug, select **Not Now**.

Configure your application

Applications for both C# and Python have been provided, as well as a sample text file you'll use to test the summarization. Both apps feature the same functionality. First, you'll complete some key parts of the application to enable it to use your Azure AI Language resource.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Labfiles/02-qna** folder and expand the **CSharp** or **Python** folder depending on your language preference and the **qna-app** folder it contains. Each folder contains the language-specific files for an app into which you're you're going to integrate Azure AI Language question answering functionality.
2. Right-click the **qna-app** folder containing your code files and open an integrated terminal. Then install the Azure AI Language question answering SDK package by running the appropriate command for your language preference:

C#:

CodeCopy

```
dotnet add package Azure.AI.Language.QuestionAnswering
```

Python:

CodeCopy

```
pip install azure-ai-language-questionanswering
```

3. In the **Explorer** pane, in the **qna-app** folder, open the configuration file for your preferred language
    - **C#**: appsettings.json
    - **Python**: .env
4. Update the configuration values to include the **endpoint** and a **key** from the Azure Language resource you created (available on the **Keys and Endpoint** page for your Azure AI Language resource in the Azure portal). The project name and deployment name for your deployed knowledge base should also be in this file.
5. Save the configuration file.

Add code to the application

Now you're ready to add the code necessary to import the required SDK libraries, establish an authenticated connection to your deployed project, and submit questions.

1. Note that the **qna-app** folder contains a code file for the client application:
    - **C#**: Program.cs
    - **Python**: qna-app.py

    Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Text Analytics SDK:

    C#: Programs.cs

    C#Copy

```
// import namespaces
using Azure;
using Azure.AI.Language.QuestionAnswering;
```

Python: qna-app.py

CodeCopy

```
# import namespaces
from azure.core.credentials import AzureKeyCredential
from azure.ai.language.questionanswering import QuestionAnsweringClient
```

2. In the **Main** function, note that code to load the Azure AI Language service endpoint and key from the configuration file has already been provided. Then find the comment **Create client using endpoint and key**, and add the following code to create a client for the Text Analysis API:

C#: Programs.cs

CodeCopy

```
// Create client using endpoint and key
AzureKeyCredential credentials = new AzureKeyCredential(aiSvcKey);
Uri endpoint = new Uri(aiSvcEndpoint);
QuestionAnsweringClient aiClient = new QuestionAnsweringClient(endpoint,
credentials);
```

Python: qna-app.py

CodeCopy

```
# Create client using endpoint and key
credential = AzureKeyCredential(ai_key)
ai_client = QuestionAnsweringClient(endpoint=ai_endpoint,
credential=credential)
```

3. In the **Main** function, find the comment **Submit a question and display the answer**, and add the following code to repeatedly read questions from the command line, submit them to the service, and display details of the answers:

C#: Programs.cs

CodeCopy

```
// Submit a question and display the answer
string user_question = "";
while (user_question.ToLower() != "quit")
    {
        Console.Write("Question: ");
        user_question = Console.ReadLine();
        QuestionAnsweringProject project = new
QuestionAnsweringProject(projectName, deploymentName);
        Response<AnswersResult> response =
aiClient.GetAnswers(user_question, project);
        foreach (KnowledgeBaseAnswer answer in response.Value.Answers)
        {
            Console.WriteLine(answer.Answer);
            Console.WriteLine($"Confidence: {answer.Confidence:P2}");
            Console.WriteLine($"Source: {answer.Source}");
            Console.WriteLine();
        }
    }
```

Python: qna-app.py

CodeCopy

```
# Submit a question and display the answer
user_question = ''
while user_question.lower() != 'quit':
    user_question = input('\nQuestion:\n')
    response = ai_client.get_answers(question=user_question,
                                     project_name=ai_project_name,
                                     deployment_name=ai_deployment_name)
    for candidate in response.answers:
        print(candidate.answer)
        print("Confidence: {}".format(candidate.confidence))
        print("Source: {}".format(candidate.source))
```

4. Save your changes and return to the integrated terminal for the **qna-app** folder, and enter the following command to run the program:
    - **C#**: `dotnet run`
    - **Python**: `python qna-app.py`

    Tip: You can use the **Maximize panel size** (^) icon in the terminal toolbar to see more of the console text.

5. When prompted, enter a question to be submitted to your question answering project; for example `What is a learning path?`.
6. Review the answer that is returned.
7. Ask more questions. When you're done, enter `quit`.

Clean up resources

If you're finished exploring the Azure AI Language service, you can delete the resources you created in this exercise. Here's how:

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. Browse to the Azure AI Language resource you created in this lab.
3. On the resource page, select **Delete** and follow the instructions to delete the resource.