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. In this exercise, you'll use the Azure AI Language Python SDK for text analytics to implement a simple question answering application.

While this exercise is based on Python, you can develop question answering applications using multiple language-specific SDKs; including:

- Azure AI Language Service Question Answering client library for Python
- Azure AI Language Service Question Answering client library for .NET

This exercise takes approximately **20** minutes.

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. Select **Create a resource**.
3. In the search field, search for **Language service**. Then, in the results, select **Create** under **Language Service**.
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 in the **Resource Management** section. 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:

    a. On the bar at the top if the page, select the **Settings (⚙)** button.
    b. On the **Settings** page, view the **Resources** tab.
    c. Select the language resource you just created, and click **Switch resource**.
    d. 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 **Select the language for all projects**, 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://learn.microsoft.com/en-us/training/support/faq?pivots=general`

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://learn.microsoft.com/en-us/training/support/faq?pivots=general`
   - **Question**: `What are the different types of modules on Microsoft Learn?`
   - **Answer**:
     `Microsoft Learn offers various types of training modules, including role-based learning paths, product-specific modules, and hands-on labs. Each module contains units with lessons and knowledge checks to help you learn at your own pace.`

3. Select **Done**.

4. In the page for the **What are the different types of modules on Microsoft Learn?** question that is created, expand **Alternate questions**. Then add the alternate question `How are training modules organized?`.

   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 module types question, expand **Follow-up prompts** and add the following follow-up prompt:

   - **Text displayed in the prompt to the user**: `Learn more about training`.
   - Select the **Create link to new pair** tab, and enter this text:
     `You can explore modules and learning paths on the [Microsoft Learn training page](https://learn.microsoft.com/training/).`
   - 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 module types 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 `What are the different types of modules on Microsoft Learn?`. The answer you created should be returned along with a follow-up prompt link.
7. Select the **Learn more about training** follow-up link. The follow-up answer with a link to the training 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 from the navigation menu on the left.

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 Cloud Shell

You'll develop your question answering app using Cloud Shell in the Azure portal. The code files for your app have been provided in a GitHub repo.

1. In the Azure Portal, use the **[>_]** button to the right of the search bar at the top of the page to create a new Cloud Shell in the Azure portal, selecting a ***PowerShell*** environment. The cloud shell provides a command line interface in a pane at the bottom of the Azure portal.

   > **Note**: If you have previously created a cloud shell that uses a *Bash* environment, switch it to ***PowerShell***.

2. In the cloud shell toolbar, in the **Settings** menu, select **Go to Classic version** (this is required to use the code editor).

   Ensure you've switched to the classic version of the cloud shell before continuing.

3. In the PowerShell pane, enter the following commands to clone the GitHub repo for this exercise:

   Code Copy

   ```
   rm -r mslearn-ai-language -f
   git clone https://github.com/microsoftlearning/mslearn-ai-language
   ```

   > **Tip**: As you enter commands into the cloudshell, the ouput may take up a large amount of the screen buffer. You can clear the screen by entering the `cls` command to make it easier to focus on each task.

4. After the repo has been cloned, navigate to the folder containing the application code files:

   Code Copy

   ```
   cd mslearn-ai-language/Labfiles/02-qna/Python/qna-app
   ```

Configure your application

1. In the command line pane, run the following command to view the code files in the **qna-app** folder:

   Code Copy

   ```
   ls -a -l
   ```

   The files include a configuration file (**.env**) and a code file (**qna-app.py**).

2. Create a Python virtual environment and install the Azure AI Language Question Answering SDK package and other required packages by running the following command:

   Code Copy

```
python -m venv labenv
./labenv/bin/Activate.ps1
pip install -r requirements.txt azure-ai-language-questionanswering
```

3. Enter the following command to edit the configuration file:

Code Copy

```
code .env
```

The file is opened in a code editor.

4. In the code file, update the configuration values it contains to reflect the **endpoint** and an authentication **key** for 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. After you've replaced the placeholders, within the code editor, use the **CTRL+S** command or **Right-click > Save** to save your changes and then use the **CTRL+Q** command or **Right-click > Quit** to close the code editor while keeping the cloud shell command line open.

Add code to user your knowledge base

1. Enter the following command to edit the application code file:

Code Copy

```
code qna-app.py
```

2. Review the existing code. You will add code to work with your knowledge base.

> **Tip**: As you add code to the code file, be sure to maintain the correct indentation.

3. In the code file, 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 Question Answering SDK:

Code Copy

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

4. 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 question answering client:

Code Copy

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

5. In the code file, 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:

Code

```
# Submit a question and display the answer
user_question = ''
while True:
    user_question = input('\nQuestion:\n')
    if user_question.lower() == "quit":
        break
    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))
```

6. Save your changes (CTRL+S), then enter the following command to run the program (you maximize the cloud shell pane and resize the panels to see more text in the command line pane):

Code

```
python qna-app.py
```

7. When prompted, enter a question to be submitted to your question answering project; for example `What is a learning path?`.
8. Review the answer that is returned.
9. 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. Close the Azure cloud shell pane
2. In the Azure portal, 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.

More information

To learn more about question answering in Azure AI Language, see the Azure AI Language documentation.