**Utilizing Power Query in Power BI to call a Text Analytics API or an LLM (Large Language Model)**

Utilizing Power Query in Power BI to call a Text Analytics API or an LLM (Large Language Model) API can be considered API integration within the Power Platform. This method involves leveraging Power BI’s capabilities to fetch and process data from external APIs, thereby integrating third-party services into your Power Platform solution.

Here’s a detailed guide on how to achieve this:

**Step-by-Step Guide for API Integration with Power Query in Power BI**

**1. Understanding the API**

1. **API Documentation**:
   * Review the API documentation to understand the available endpoints, request formats, required parameters, authentication methods, and response formats.
2. **API Authentication**:
   * Obtain necessary credentials such as API keys, tokens, or client IDs/Secrets. Ensure you have the permissions to access the API.

**2. Setting Up Power Query in Power BI**

1. **Open Power BI Desktop**:
   * Open Power BI Desktop and go to the Home tab.
2. **Get Data**:
   * Click on “Get Data” and select “Blank Query” from the options.

**3. Using Power Query Editor**

1. **Open Advanced Editor**:
   * In the Power Query Editor, open the Advanced Editor to write your custom M code for calling the API.
2. **M Code for API Call**:
   * Write the M code to call the Text Analytics API or LLM API.Note that to access Text Analytics ,you have to have Azure Cognitive Service
   * Here’s a basic example:

m

Copy code

let

// Define the API endpoint URL

apiUrl = "https://api.textanalytics.com/v3.0/sentiment",

// Set up the request headers

headers = [

#"Ocp-Apim-Subscription-Key" = "YOUR\_API\_KEY",

#"Content-Type" = "application/json"

],

// Define the request body

requestBody = Text.ToBinary("{""documents"":[{""id"":""1"",""text"":""Power BI is an amazing tool!""}]}"),

// Make the API call

apiResponse = Web.Contents(apiUrl, [

Headers = headers,

Content = requestBody,

ManualStatusHandling = {400, 401, 403, 404, 500}

]),

// Parse the JSON response

responseJson = Json.Document(apiResponse)

in

responseJson

1. **Transform and Load Data**:
   * After fetching the data from the API, you can use Power Query’s transformation capabilities to shape the data as needed. For instance, you might extract specific fields or aggregate the data.

m

Copy code

let

// Continue from the previous example

sentimentScore = responseJson[documents]{0}[score]

in

sentimentScore

1. **Error Handling**:
   * Implement error handling to manage API call failures or invalid responses. You can use conditional logic to check the status code and handle errors gracefully.

m

Copy code

let

// Previous steps

apiResponse = try Web.Contents(apiUrl, [

Headers = headers,

Content = requestBody,

ManualStatusHandling = {400, 401, 403, 404, 500}

]),

// Check for errors

responseCheck = if apiResponse[HasError] then error apiResponse[Error] else Json.Document(apiResponse[Result]),

// Process the response

sentimentScore = responseCheck[documents]{0}[score]

in

sentimentScore

**4. Using the API Data in Power BI**

1. **Create Visualizations**:
   * Use the data retrieved from the API to create meaningful visualizations in Power BI. For example, you can display sentiment scores, categorize text, or show insights derived from the LLM API.
2. **Refresh Data**:
   * Set up a data refresh schedule to ensure that your Power BI reports always have the latest data from the API.
3. **Publish to Power BI Service**:
   * Publish your Power BI report to the Power BI service to share with others. Ensure the API credentials are managed securely, especially for scheduled refreshes.

**Conclusion**

Using Power Query in Power BI to call external APIs like Text Analytics or LLM APIs is a practical example of API integration within the Power Platform. This approach allows you to enrich your Power BI reports with external data and insights, leveraging the capabilities of external services seamlessly within the Power Platform ecosystem.

By following the steps outlined above, you can effectively integrate and utilize third-party APIs in your Power BI projects, thereby enhancing the analytical capabilities and value of your Power Platform solutions.