Steps to develop POC using Open AI

To set up a local environment for developing proof-of-concepts (POCs) using OpenAI in both C# and Python, you can follow these steps:

1. Install Python: Install Python on your local machine. You can download the latest version of Python from the official Python website (<https://www.python.org/downloads/>) and follow the installation instructions specific to your operating system.
2. Install C#: If you're planning to develop POCs using C#, ensure that you have the .NET Framework installed. You can download the .NET Framework from the official Microsoft website (<https://dotnet.microsoft.com/download>) and follow the installation instructions specific to your operating system.
3. Set up a virtual environment (Python): It's good practice to use virtual environments in Python to isolate your project dependencies. You can create a virtual environment using the following commands in your command prompt or terminal:

bashCopy code

python -m venv myenv # Create a new virtual environment

Activate the virtual environment:

* For Windows:

bashCopy code

myenv\Scripts\activate

* For Linux/Mac:

bashCopy code

source myenv/bin/activate

1. Install OpenAI Python library: In your activated virtual environment, install the OpenAI Python library using pip:

bashCopy code

pip install openai

pip install openai

1. Set up C# project (Visual Studio): Open Visual Studio and create a new C# project. Ensure that you have the necessary OpenAI NuGet package installed. You can manage NuGet packages in Visual Studio by right-clicking on the project in the Solution Explorer and selecting "Manage NuGet Packages".
2. Obtain API credentials: Sign in to the OpenAI platform and obtain your API key. This key will be required for authentication when making API requests.
3. Write code and make API requests: In both C# and Python, you can now write code to interact with the OpenAI API. Here are example code snippets for making API requests:

C# Example:

csharp code

using System;

using OpenAI;

class Program

{

static void Main()

{

OpenAIConfiguration.ApiKey = "YOUR\_API\_KEY";

var prompt = "What is the capital of France?";

var response = Completion.Create(engine: "text-davinci-003", prompt: prompt, maxTokens: 50, n: 1);

Console.WriteLine(response.Choices[0].Text.Trim());

}

}

Python Example:

Python code

import openai

openai.api\_key = 'YOUR\_API\_KEY'

prompt = "What is the capital of France?"

response = openai.Completion.create(engine='text-davinci-003', prompt=prompt, max\_tokens=50, n=1)

print(response.choices[0].text.strip())

Replace **'YOUR\_API\_KEY'** with the actual API key you obtained from OpenAI.

Remember to handle exceptions, manage API rate limits, and follow best practices while developing your POCs.

Please note that the OpenAI API and libraries may have evolved since my knowledge cutoff in September 2021. I recommend referring to the OpenAI documentation and resources for the most up-to-date information on using OpenAI services in your C# and Python projects.



give me sample code in C# to create a POC for Document search for some documents using Open AI to query on those documents

ChatGPT

Certainly! Here's an example of C# code that demonstrates a proof-of-concept (POC) for document search using OpenAI to query a collection of documents:

csharpCopy code

using System;

using System.Collections.Generic;

using OpenAI;

class Program

{

static void Main()

{

OpenAIConfiguration.ApiKey = "YOUR\_API\_KEY";

// Collection of documents

List<string> documents = new List<string>

{

"Document 1: This is the first document in the collection.",

"Document 2: Here's the second document for searching.",

"Document 3: The third document contains relevant information.",

"Document 4: This is another document to be included."

};

// User query

string query = "search relevant information";

// Combine documents and query for searching

string searchQuery = query + "\n\n" + string.Join("\n\n", documents);

// Call the OpenAI API for document search

var response = OpenAI.DocumentSearch.Create(searchModel: "davinci", documents: new List<string> { searchQuery });

// Retrieve the top matching document

var topResult = response.Data[0];

Console.WriteLine($"Top Matching Document Index: {topResult.DocumentIndex}");

Console.WriteLine($"Score: {topResult.Score}");

Console.WriteLine($"Text: {documents[topResult.DocumentIndex]}");

}

}

Replace **'YOUR\_API\_KEY'** with your actual OpenAI API key. In this example, the **documents** list contains a collection of sample documents, and the **query** variable holds the user's search query. The code combines the documents and query, then makes a document search API request using the OpenAI C# library. Finally, it displays the top matching document's index, score, and text.

Ensure that you have the OpenAI NuGet package installed in your C# project. You can manage NuGet packages in Visual Studio by right-clicking on the project in the Solution Explorer and selecting "Manage NuGet Packages."

Remember to handle exceptions, manage API rate limits, and follow best practices while developing your document search application.