Backend Script Documentation - Website Crawling and OpenAI Fine-tuning Integration

1. Introduction:

The backend script described in this document serves the purpose of automating the process of website crawling and scraping using Selenium, and subsequently utilizing the scraped information to train a model through the OpenAI fine-tuning API. This comprehensive guide will provide an in-depth understanding of the script's functionality, components, and overall workflow.

2. Script Workflow Overview:

The backend script comprises several interconnected components that work cohesively to achieve the desired tasks. The general workflow is as follows:

a. Website Crawling:

- Utilizes Selenium, a powerful web scraping tool, to navigate through the Cochrane website.
- Gathers relevant information by interacting with the website's elements, such as clicking buttons, or extracting specific HTML elements.
- Extracted information is stored in memory for further processing.

b. Scraping Information:

- Utilizes Selenium's capabilities to extract desired data from the Cochrane website.
- Extracted data may include article titles, authors, abstracts, publication dates, and other relevant information.
- The scraped data is stored in appropriate data structures, such as lists or dictionaries.

c. Preparing Data for Fine-tuning:

- The scraped data is processed and transformed into a suitable format for training the model.

- Data preprocessing techniques like tokenization, cleaning, and normalization are applied to ensure consistency and quality.

- The transformed data is then organized into appropriate datasets, such as training, validation, and testing sets.

d. Model Training using OpenAI Fine-tuning API:

- The preprocessed data is sent to the OpenAI fine-tuning API, an interface for training language models.

- The script interacts with the API, providing the necessary input data, model configuration, and fine-tuning parameters.

- The API performs the actual model training process, using the provided data to fine-tune the language model.

- The fine-tuned model is saved for future use.

3. Component Details:

a. Selenium Integration:

- Selenium is a widely-used web scraping tool that automates interactions with websites.

- The script utilizes Selenium's WebDriver functionality to control a web browser.

- A compatible web driver, such as ChromeDriver, is used to launch a browser instance.

- The script interacts with the browser by locating and manipulating web elements based on their HTML attributes, XPath, or CSS selectors.

b. Website Crawling:

- The script is designed to crawl the Cochrane website, which is a trusted source for medical research.

- Specific URLs or search queries can be provided to target desired pages or sections of the website.

- Selenium is used to navigate through the website's pages, interacting with buttons, forms, and other elements as required.

- The script collects data from the website by extracting relevant information from HTML elements.

c. Data Preprocessing:

- Once the necessary data is extracted, it undergoes a preprocessing phase to prepare it for model training.

- Text data may be tokenized into words or subwords, ensuring a consistent representation.

- Additional preprocessing steps include removing stopwords, performing stemming or lemmatization, and handling special characters or formatting issues.

d. OpenAI Fine-tuning API Integration:

- The script connects to the OpenAI fine-tuning API to utilize its capabilities for model training.

- Appropriate API credentials, including the API key, are required for authentication and access.

- The script interacts with the API by sending requests with the preprocessed data and other relevant parameters.

- The API responds by performing the fine-tuning process, training the language model based on the provided data and configurations.

4. Conclusion:

The backend script detailed in this document showcases the integration of Selenium for website crawling and OpenAI's fine-tuning API for model training. By automating the process of data extraction and utilizing the OpenAI API's powerful capabilities, the script provides an efficient solution for scraping information from the Cochrane website and training models for further analysis or application. Understanding the various components and workflow of the script will help in maintaining, expanding, and customizing its functionality as per specific requirements.\

1. Tech Stack used for the Web App

- Flask
  (Flask is a lightweight web framework for Python. It is designed to be simple and easy to use, allowing developers to quickly build web applications and APIs. Flask follows the Model-View-Controller (MVC) architectural pattern, but it does not enforce any particular implementation or structure. This flexibility makes it a popular choice for building small to medium-sized web applications.)
- OpenAI (for modeling purposes)
- HTML, CSS (for designing purposes)

2. List of APIs that are already available

- OpenAI Completions API
- OpenAI Embeddings API
- OpenAI Fine tuning API

3. Modules that are currently functioning on the Website

- Search Module (As shown in screenshots previously shared)

4. Link to the currently functioning Website OR functioning Video of the Website

- Link will be generated once website is deployed on AWS.

5. Integrations done with the Backend

- APIs mentioned above have been integrated in the backend