Team Whackos

Abhijeet Kumar
Aryan Mittal
Devansh Verma
Riya Sanket Kashive

Gen AI PS by Overlayy

Documentation

Overview

This solution aims to automate the extraction and content analysis of all embedded links from a website, regardless of their location, and includes asking concise and relevant questions, the most relevant links and topics for those questions, all complete with an automated verification and metric system for assessing their aforementioned parameters (conciseness and relevance).

We have used Selenium for data scraping, JSON files for data storage, and the duckduckgo_search library with the gemini API for question generation. For mapping links to questions and as a relevance metric, we have used TFIDF Vectorisation.

Goals

Problem Statement:
https://docs.google.com/document/d/1GaQza95lxQJHXrCtrdntavKL3a0_Bg_Kvj3QmZuDNWA/edit

Specifications

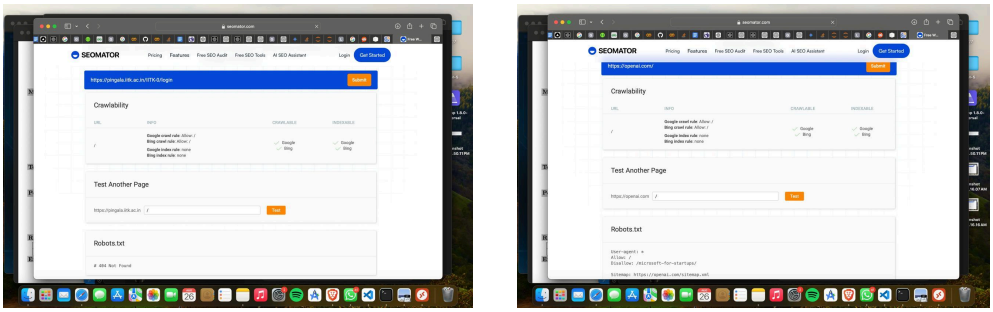
While this was a sprint of a hackathon, and there were time constraints, we were careful to make deliberate choices in order to come to a general solution capable of analysing websites which are widely known and used, as well as those that are not. Here are the specifications of the technology that we used to build our solution, followed by a brief explanation for choosing them.

Scraping Links Using Selenium:

Selenium was used over traditional web crawler APIs, which gave us an edge over others in handling dynamic and complex web pages. It detects links regardless of location or format (text, buttons, or secured sections), something a crawler API might not be able to do. A crawler typically works in two ways: using beautifulsoup/scrapy to do HTML parsing,, where it cannot catch links embedded in buttons or secured websites. The other way is by accessing the site map made by a web browser, which, for sites that do not encounter much traffic, is not extensive and thus unavailable. Selenium with its locator strategies, handles these situations with ease.

Cases in point:

1. openai.com, the website used in the example itself, is a secure website where HTML parsing is limited and cannot catch all of the links embedded. Selenium is not held back by this limitation.
2. pingala@iitk.ac.in is a website of IITK for IITK to manage academics, admissions and fee payment. Thus it gathers negligible traction outside the community. For such a website, a site map is unavailable and the crawler might fail.



Sitemap not displayed for pingala@iitk.ac.in (left), but is for openai.com(right)

Storing Links in JSON Format:

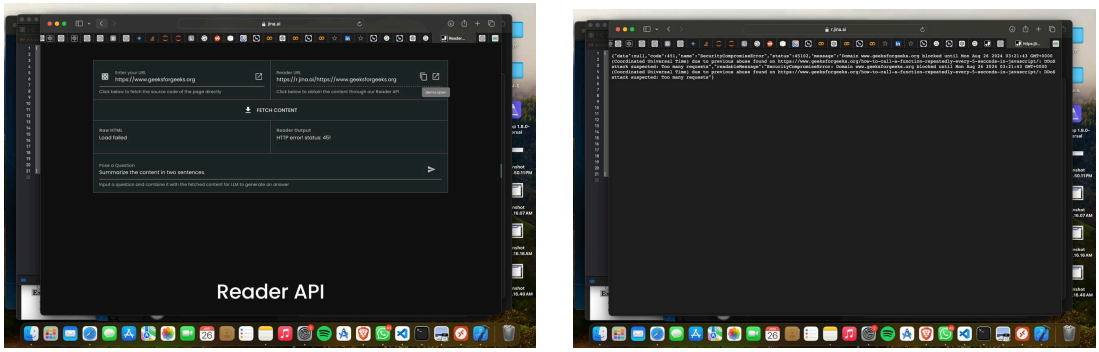
A JSON file was chosen for its simplicity and ease of integration with other tools and programming languages. One of the JSON files contains a list of all the extracted links, categorized based on their source (text, button, secured section, etc.), and the second

one contains the questions generated along with their most relevant links and their topics.

Question generation using duckduckgo_search and Gemini API:

The duckduckgo_search library was used to query each link, and was chosen over LLM models for retrieval of legitimate, factual information about each extracted link. It is also the only search engine key that is not paid. Choosing a search engine library/ key is particularly important for obtaining reliable information about the website content for content generation as opposed to an LLM model, since the latter work such that they can only provide approximate or speculative responses. LLM models might be able to give correct information when it comes to well-known sites, but for sites that do not face much traffic, there is a fair chance that the LLM model is not trained on that site, or not trained well-enough on that topic, to provide legitimate answers, or even ask relevant questions. Jina AI was not used in spite of being suggested, due to their limitations when it came to secured websites.

Case in point:



The result after using Jina AI for geeksforgeeks.com.

In combination with the duckduckgo_search for obtaining factual information, we used the Gemini API for formatting it to a search prompt, and then parsing the results obtained on search for question generation.

TFIDF Vectorisation for selecting the most relevant links:

TFIDF is a measure of a word's originality by comparing how many times it appears in a document to how many documents it appears in. Based on the bag of words model that creates a matrix containing information about the most and least relevant words in a document, TFIDF is particularly useful in topic modeling.

Milestones

Our solution achieved an accuracy of 83%.