

Web Search Summarizer



November 4, 2024

Himakar Raju Gunda

Code Analysis and Documentation for Automated Web Scraping and AI Response Generation

Date: November 04, 2024

Prepared by: Himakar Raju Gunda

This document provides a comprehensive analysis of a script designed for automated data extraction from web sources, AI-driven response generation, and structured report creation in Word documents.

Table of Contents

[Table of Contents 2](#_Toc181600430)

[1. Introduction 3](#_Toc181600431)

[2. Code Analysis and Explanations 3](#_Toc181600432)

[measure\_time(func) 3](#_Toc181600433)

[load\_model() 3](#_Toc181600434)

[generate\_response(prompt, model, tokenizer, device) 3](#_Toc181600435)

[extract\_text\_from\_url(url) 3](#_Toc181600436)

[scrape\_google\_search(query, num\_results=10) 3](#_Toc181600437)

[get\_user\_prompt\_choice() 3](#_Toc181600438)

[generate\_prompt(choice, search\_results) 4](#_Toc181600439)

[create\_docx(query, response, docx\_filename) 4](#_Toc181600440)

[main() 4](#_Toc181600441)

[3. Future Extensions and Improvements 4](#_Toc181600442)

[4. Conclusion 4](#_Toc181600443)

[5. Appendix 4](#_Toc181600444)

# Introduction

**Objective:**

This code enables automated data extraction, AI-based summarization, and structured report generation, primarily for use cases such as market analysis, competitor research, and automated knowledge summarization.

**Functionality:**

* **Data Extraction:** Scrapes web content based on user queries.
* **AI Response Generation:** Uses transformer models to provide insightful responses.
* **Report Creation:** Formats and exports data in a Word document format.

**Prerequisites:**

**Required Libraries:**

torch, transformers, requests, BeautifulSoup, docx.

**Hardware:**

GPU for optimal performance if working with large transformer models.

# Code Analysis and Explanations

## measure\_time(func)

**Purpose:** A decorator to calculate and print the execution time of a function.

**Inputs:** Accepts a function with varying arguments (\*args, \*\*kwargs).

**Outputs:** Returns the original function output and prints execution time.

## load\_model()

**Purpose:** Loads a specified transformer model and tokenizer.

**Inputs:** None, uses configured model path.

**Outputs:** Returns the model, tokenizer, and device (CPU/GPU).

**Explanation:** Checks for GPU availability for efficient processing.

## generate\_response(prompt, model, tokenizer, device)

**Purpose:** Generates text responses based on a user prompt.

## extract\_text\_from\_url(url)

**Purpose:** Extracts main content from a URL and cleans it for processing.

## scrape\_google\_search(query, num\_results=10)

**Purpose:** Fetches and parses Google search results for a query.

## get\_user\_prompt\_choice()

**Purpose:** Prompts the user to select a query format.

## generate\_prompt(choice, search\_results)

**Purpose:** Generates a prompt based on user’s selection and results.

## create\_docx(query, response, docx\_filename)

**Purpose:** Creates a formatted .docx document for the query and AI response.

## main()

**Purpose:** Runs the main program loop, integrating all components for end-to-end processing.

# Future Extensions and Improvements

**Scalability Enhancements:** Parallel processing and caching to handle larger volumes efficiently.

**Optimization Suggestions:** Use further quantization and asynchronous scraping.

**Feature Expansion:** Include options for customizable report templates and multilingual support.

# Conclusion

The code enables efficient, automated data gathering and report generation. With extensibility and a modular design, it can adapt to diverse automated reporting needs, with potential scalability improvements.

# Appendix

**Dependencies:**

torch, transformers, requests, BeautifulSoup, docx.

**References and Documentation:**

See library documentation for ‘transformers’ and ‘torch’ for further configuration details.