Brief Explanation of the Approach and Design Decisions

Overview

The research assistant is designed to automatically generate meaningful answers to user questions by combining natural language processing (NLP) techniques with web search capabilities. The core idea is to transform a broad research question into targeted search queries, gather relevant information using an external search API, and synthesize the findings into a coherent answer with source citations.

Approach

1. **Query Decomposition:**  
   The system first breaks down a user’s research question into one or several effective search queries. This is done by generating keywords and removing less informative words to ensure that the search queries capture the essence of the original question.
2. **Information Gathering:**  
   The generated search queries are then used with the Google Custom Search API. This API retrieves web search results which include titles, snippets, and URLs. The intention is to leverage real-world data to provide up-to-date and relevant information.
3. **Answer Synthesis:**  
   The results from the search API are concatenated into a context string. Using a natural language generation model (in our case, the GPT-3.5-turbo model accessed via OpenAI's ChatCompletion endpoint), the system synthesizes these disparate pieces of information into a single, coherent answer. The answer is formatted to include in-text citations corresponding to the search results.
4. **Output and Citation:**  
   Finally, the assistant outputs both the synthesized answer and a list of source links, ensuring that users know where the information originated.

Design Decisions

1. Modularity and Separation of Concerns

* **Function-Based Modular Design:**  
  Each step (query generation, search, synthesis) is encapsulated in its own function:
  + break\_down\_question: Generates search queries.
  + perform\_search: Fetches results using the Google Custom Search API.
  + synthesize\_answer: Creates a summarized answer from the search results.  
    This separation allows the system to be easier to understand, maintain, and extend.
* **Error Handling:**  
  The code incorporates error detection for API communication and missing inputs, ensuring that the script aborts gracefully if necessary data (such as API keys or search results) is not available.

2. API Integration

* **OpenAI for NLP Tasks:**  
  The decision to use OpenAI’s GPT-3.5-turbo model for both query decomposition and answer synthesis leverages a powerful, state-of-the-art language model. This simplifies the processing pipeline since a single model (with different prompts) can handle multiple NLP tasks.
* **Google Custom Search API for Real-World Information:**  
  By integrating with the Google Custom Search API, the system obtains dynamic, real-world content. This ensures that the synthesized answers are grounded in up-to-date information from across the web.

3. Scalability and Flexibility

* **Search Query Generation:**  
  The approach of generating search queries by filtering out stop words is simple but effective. It allows the system to be fast and lightweight, while still enabling the capture of essential keywords. Future iterations may incorporate more advanced NLP techniques if needed.
* **Structured Result Formatting:**  
  The search results are transformed into a uniform structure (with title, snippet, and link), which makes it straightforward to assemble the context for answer synthesis. This design decision helps in integrating results regardless of minor differences in API response formats.

4. User Experience

* **Simplicity and Clarity:**  
  The script prints out key stages of processing (like the generated search queries and the final synthesized answer). This transparency aids users in understanding how their question is transformed and where the provided information comes from.
* **Citation of Sources:**  
  Including a list of source URLs in the final output ensures traceability and builds user trust. Users can verify the information or explore further if they desire.

Conclusion

The design of this research assistant emphasizes a modular processing pipeline that starts from transforming a broad query into crisp, targeted search queries, retrieving relevant data using a robust search API, and finally synthesizing the gathered information using a state-of-the-art language model. The decisions were guided by the need for clarity, maintainability, and a seamless integration of NLP and web search, creating a tool that is both practical and user-friendly.