HOW IT WORKS?

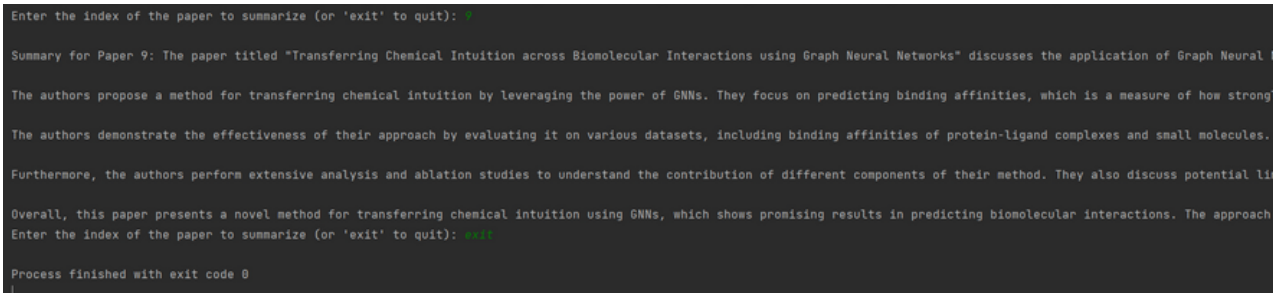
```
C:\Users\iamsa\PycharmProjects\myproject\venv\Scripts\python.exe C:\Users\iamsa\AppData\Roaming\JetBrains\PyCharmCE2022.3\scratches\improve.py
Enter the topic for research papers: LLMs
Enter the timeframe (e.g., 'Within 1 Month'): Within 3 Months
```

Step 1: User Input And Query Generation: The user is asked to input the topic on which he wants to read the research papers. There can be as many topics as there exist on the web for which the research papers can be fetched. After receiving the initial input, the user is then asked to enter a timeframe as the research papers will be gathered from this time period. Both inputs are used by a function named generate_openai_query() that is used in the generation of the search query based on the user's topic and timeframe. This is important as it tailors the query to the user's input, ensuring the generated query is relevant and specific to the user's research interest.

```
List of Research Papers:
1. Legal-Tech Open Diaries: Lesson learned on how to develop and deploy light-weight models in the era of humongous Language Models
   URL: https://arxiv.org/abs/2210.13086

2. GPT Takes the Bar Exam
   URL: https://arxiv.org/abs/2212.14402

3. Large Language Models as Tax Attorneys: A Case Study in Legal Capabilities Emergence
   URL: https://arxiv.org/abs/2306.07075

4. A Short Survey of Viewing Large Language Models in Legal Aspect
   URL: https://arxiv.org/abs/2303.09136

5. Law Article-Enhanced Legal Case Matching: a Model-Agnostic Causal Learning Approach
   URL: https://arxiv.org/abs/2210.11012

6. LegalMFiT: Efficient Short Legal Text Classification with LSTM Language Model Pre-Training
   URL: https://arxiv.org/abs/2109.00993

7. How Smart are Smart Readers? LLMs and the Future of the No-Reading Problem
   URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4491043

8. Legal Prompting: Teaching a Language Model to Think Like a Lawyer
   URL: https://arxiv.org/abs/2212.01326

9. Parameter-Efficient Legal Domain Adaptation
   URL: https://arxiv.org/abs/2210.13712

10. Legal Prompt Engineering for Multilingual Legal Judgement Prediction
   URL: https://arxiv.org/abs/2212.02199
```

Step 2: Research Paper Search And Display: In the second step, the program uses the Metaphor API to search for research papers based on the generated search query from the previous step. The Metaphor API is accessed using the Metaphor class provided by the metaphor_python package. The search function of the Metaphor API is invoked, providing the generated search query. The Metaphor API searches for recent research papers related to the specified topic and timeframe. The search results, including paper titles and URLs, are retrieved and stored for further use. To display the results in a user-friendly manner, the program uses the display_paper_list_with_indexes function. This function formats and prints the list of research

papers, including their titles and corresponding URLs, making it easy for the user to explore the search results.



Step 3: Summarizing a Selected Research Paper: In the final step, the program allows the user to select a research paper from the displayed list for summarization. The user is prompted to enter the index of the paper they want to summarize. The program validates the user's input and, if valid, uses the get_paper_summary function to generate a summary for the selected research paper. The get_paper_summary function uses the GPT-3.5 Turbo model through the ChatCompletion.create function to generate a summary based on the content of the selected paper. The resulting summary is then displayed using the display_paper_summary function, which formats the summary to ensure a maximum of 100 characters per line, making it more readable and aesthetically pleasing for the user.