International Journal of Advanced Smart Convergence Vol.15 No.1 1-9 (2026)
http://dx.doi.org/10.7236/IJASC.2026.15.1.1

IJASC 26-1-1

QA Maker: Automatic Generation of Domain-specific QA System utilizing GraphRAG

Jiyoon Ok, Juyeon Soung, Minseo Kim, Kitae Hwang*

Student, Department of Computer Engineering, Hansung University, Korea
** Professor, Department of Computer Engineering, Hansung University, Korea*

*ojoj8ojo@icloud.com, tjdwndus1325@gmail.com, kimalstj2739@naver.com, calafk@hansung.ac.kr**

Abstract

Today's Internet search systems, searching the entire world's web, boast outstanding performance and inference results. On the other hand, search systems built within organizations that operate domains, such as universities, corporations, and public institutions, rely on keyword matching methods, resulting in very low performance, with searches failing or returning inappropriate answers that only match keywords regardless of the intent of the query. Moreover, building a superior search system requires enormous investment. This paper proposes QA Maker, a system that automatically collects all subdomain web pages and data by entering a representative URL for a specific domain, constructs a knowledge graph, and utilizes the GraphRAG technique to automatically generate a QA(Question-answering) system with high search and inference capabilities. Experiments across two test institution showed that while each institution's existing search system achieved an accuracy of approximately 19%, the QA system generated by QA Maker achieved an accuracy of approximately 90%. In conclusion, we believe that QA Maker, proposed in this paper, is an excellent system that can easily build high-performance QA systems at low cost across a variety of domains.

Keywords: *Question-Answering, GraphRAG, Knowledge Graph, LLM, QA System, Graph DB*

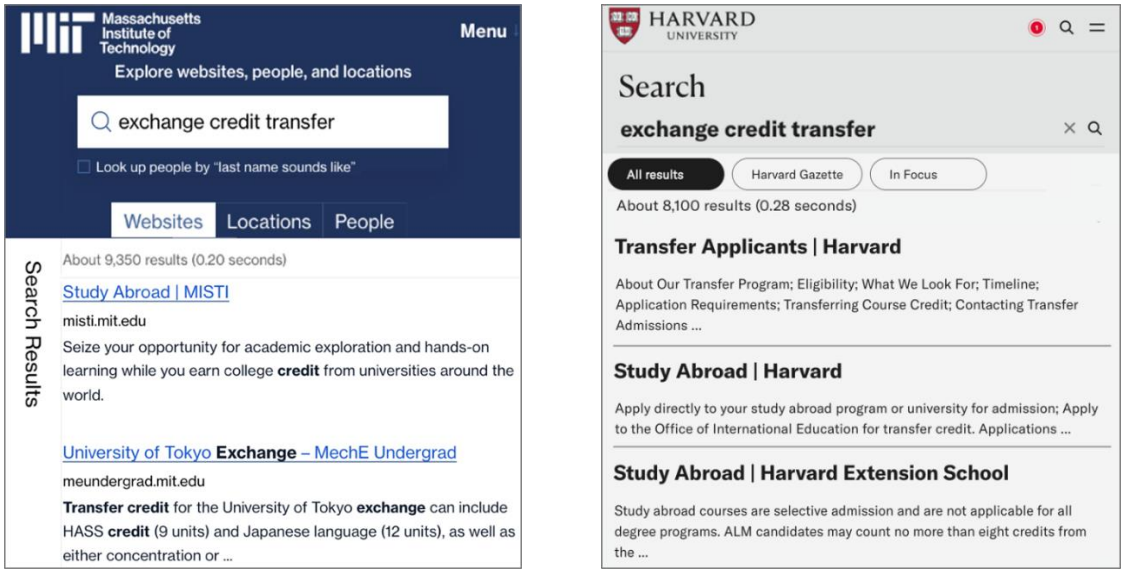
1. Introduction

Internet search systems like Naver, Google, and Bing leverage massive capital and technological prowess to search the entire world's web, demonstrating exceptionally high search performance [1]. These search systems are evolving into AI-based question-answering systems that leverage LLM, trained on vast amounts of data from the Internet, to understand the context of complex queries and generate inferential answers [2].

Meanwhile, most institutions, including universities, corporations, and public institutions, have search systems that enable internal information retrieval. However, these systems rely on keyword matching, which

Manuscript Received: January. 15, 2026 / Revised: January. 30, 2026 / Accepted: February. 1, 2026
Corresponding Author: calafk@hansung.ac.kr
Tel: +82-2-760-4300, Fax: +82-2-760-5771
Professor, School of Computer Engineering, Hansung University, Korea

can lead to searches failing if keywords don't match or returning irrelevant results. Furthermore, they lack the ability to provide answers through natural language queries or inference, resulting in virtually no use of these systems. Figure 1 shows the results of a search for "exchange credit transfer" in the search systems of MIT and Harvard. MIT only displayed results matching the keywords "exchange" and "credit," regardless of context. Harvard displayed completely irrelevant results, such as "transfer information," at the top of the search results.



(a) Search in MIT Website **(b) Search in Harvard University Website**

Figure 1. Examples of poor search performance in domains of MIT and Harvard University

This clearly demonstrates the inherent limitations of current domain-specific search systems based on keyword matching. An improvement over keyword matching is the vector database method, which stores data in a vector database and allows for searching for similar terms [3]. However, this method requires high development costs and does not significantly improve search quality. Furthermore, it has limitations in implementing sophisticated search systems, such as natural language search or inference search. Meanwhile, there is a method of reconstructing a high-level search system by learning domain-specific information using SLM, but this requires astronomical development and ongoing maintenance costs.

To address this issue, this paper proposes QA Maker, a technique for automatically generating a high-quality QA(question-answering) system for a specific domain. We present the development of QA Maker as a web framework. QA Maker accepts only the representative URL of a specific domain, automatically collects and analyzes all web pages and all linked documents within the domain, and creates a unified knowledge graph and knowledge database. Furthermore, the paper provides a web QA system that receives user queries and generates responses using this knowledge database. When the QA Maker creates a knowledge graph DB, GraphRAG technology is used. GraphRAG technology is known to provide not only accurate answers to queries but also high-quality answers to inferential queries [4]. This QA system allows natural language queries, presents supporting documents and URLs for answers, and provides accuracy for answers calculated based on the RAGAS framework [5], providing users with confidence in the answers.

To demonstrate the superiority of the proposed technique, we selected two universities in Korea and compared the query response quality of their existing search systems with that of a QA system generated using

QA Maker. The comparison results showed that the response accuracy of the existing search systems at each university did not exceed 20 out of 100, while the generated QA system achieved a response accuracy of over 90 out of 100.

2. Related Works

2.1 LLM-based question-answering system and RAG

QA(Question-answering) systems automatically provide answers to questions posed by users in natural language, and have evolved since the early 1960s with the BASEBALL and LUNAR systems [6]. Currently, most companies and universities rely on keyword matching methods, which allows searches only when exact keywords are entered, and have structural limitations such as search failures or inappropriate results when keywords do not match. Following this, LLMs like GPT-4, Claude, and Gemini emerged, and QA systems utilizing these systems learned from massive text data, overcoming the limitations of keyword matching and achieving high response accuracy. However, LLMs still exhibit issues such as inaccurate answers due to hallucination, a lack of up-to-date information, and limitations in domain-specific knowledge such as law, medicine, and finance [7]. To overcome these limitations, the Responsive Aggregation (RAG) technique was introduced [8]. RAG refines LLM output by referencing external knowledge bases, thereby mitigating the problems of hallucination, lack of domain-specific knowledge, and outdated information. However, traditional RAG still has limitations when dealing with questions requiring complex relationships between documents or multi-level inference.

2.2 Knowledge Graph and GraphRAG

A knowledge graph is a semantic network that represents the complex relationships between real-world entities in a structured framework. It is a key technology that transforms information retrieval from simple string matching to sophisticated entity-based search. Since Google introduced the Knowledge Graph in 2012, it has been utilized in various fields, including search engines, recommendation systems, and QA systems [9]. A representative example is the "Visualize Me By Photo" system, which builds a knowledge graph from the massive personal data stored on smartphones [10]. Meanwhile, the GraphRAG technique has recently emerged to overcome the limitations of traditional RAG. GraphRAG constructs a connected knowledge network by graphing key concepts and relationships within documents. It goes beyond the simple vector similarity-based retrieval of traditional RAGs and leverages complex relationships between entities to enable more precise and comprehensive retrieval and inference. Proposed by Microsoft, GraphRAG supports both local and global searches, accurately processing both detailed queries focused on specific entities and comprehensive queries across the entire dataset [11]. In a previous study prior to this one, our research team implemented and evaluated an EPUB reader using GraphRAG, and achieved a high accuracy of approximately 90% in both simple fact retrieval and inference-based retrieval [12].

3. QA Maker

This section describes the structure and core technologies of QA Maker.

3.1 System Overview of QA Maker

The process by which QA Maker automatically creates a QA system for a specific domain consists of three steps, as shown in Figure 2. In the first step, QA Maker automatically collects all sub-web pages and attached

documents through web crawling based on the domain URL entered by the administrator. It then identifies the hierarchical relationships between web pages and creates a file containing these hierarchical relationships.

The second step is structuring. It removes unnecessary data from each web page or document within the domain, reflects the hierarchical relationships contained in the hierarchical relationship files, and stores them in a structure that facilitates the creation of a knowledge graph. In the third step, QA Maker reads the structured files, generates a knowledge graph from them, and builds the knowledge graph into a graph database. The pre-built web server and knowledge database become the QA system.

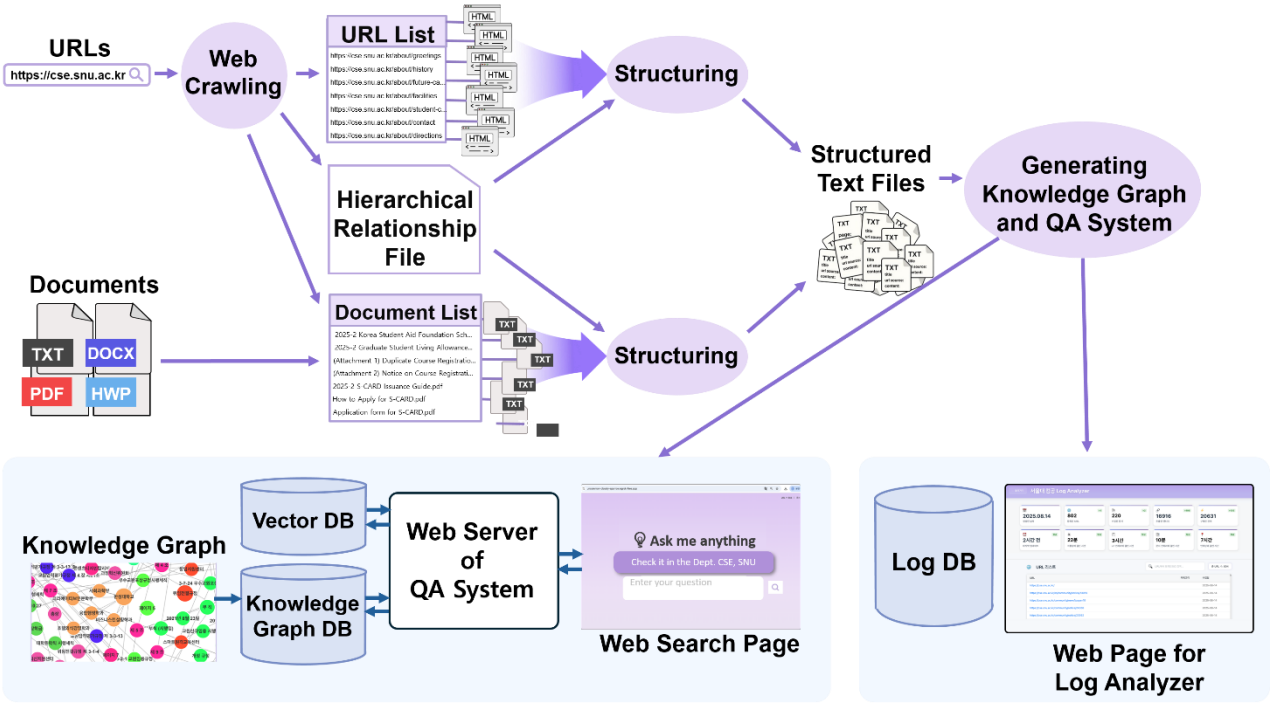


Figure 2. The Workflow of QA Maker

3.2 Web Crawling

QA Maker uses web crawling to collect all sub-web pages and attached documents for the entered domain URL in a tree-like hierarchical structure. Web crawling utilizes the DFS algorithm to store all web pages and attached document URLs in the domain as lists. Simultaneously, the hierarchical relationships of the website are represented in a JSON-formatted breadcrumb structure and stored as a hierarchical file, as shown in Figure 3. Figure 3 shows an example of a hierarchical relationship file generated by crawling the Dept of CSE(Computer Science and Engineering) at the Seoul National University. The hierarchical structure information stored in the hierarchical relationship file is used to enhance the context of each page during the structuring phase.

```
{
  "url": "https://cse.snu.ac.kr/",
  "depth": 0,
  "page_title": "SNU Computer Science and Engineering",
  "breadcrumb": "SNU Computer Science and Engineering",
  "is_document": false,
  "children_count": 57,
  "children": [
    {
      "url": "https://cse.snu.ac.kr/research/labs",
      "depth": 1,
      "page_title": "Research Laboratories | SNU",
      "breadcrumb": "SNU Computer Science and Engineering / Research Laboratories | SNU
      "is_document": false,
      "children_count": 95,
      "children": [
        {
          "url": "https://cse.snu.ac.kr/api/v1/file/171105162296342_AI_Human_Interactior
          "depth": 2,
          "page_title": "Intelligent Software Lab",
          "breadcrumb": "SNU Computer Science and Engineering / Research Laboratories |
          "is_document": true,
          "children_count": 8,
          "children": []
        }
```


Figure 3. An Example of breadcrumb JSON for website of CSE at SNU

3.2 Structuring

Structuring is the process of converting each web page and document collected through crawling into Markdown-formatted text, reflecting the information in the hierarchical structure file, and saving it as a text file. This is crucial for creating an accurate knowledge graph. For example, if knowledge is generated solely based on the information presented on a single web page without reflecting the hierarchical structure between web pages, the resulting knowledge graph will be less efficient for retrieval. For example, if there is a professor introduction page under the department introduction homepage, simply extracting professor information from the professor introduction web page may result in missing information about the professor's department.

In this paper, each web page and document is structured as a text document in Markdown format [13]. The process of structuring crawled web pages into Markdown-formatted text files is rule-based. General web pages are converted to Markdown format based on rules established by the Jina API, while web pages within bulletin boards are processed using the rules developed in this study. For general web pages, we focus on removing unnecessary elements like headers, footers, and sidebars. For bulletin board web pages, we specialize in accurately extracting structured information like the title, author, creation date, and body text. This process removes unnecessary HTML tags, scripts, and style elements, extracting only meaningful structural elements like titles, lists, and tables.

3.4 Generating QA System

The QA system generated by QA Maker consists of three elements: a knowledge graph database, a web system that receives user queries and outputs responses, and log data and a log output web page that records the QA system's creation process. These can be viewed through the Log Analyzer menu in QA Maker.

The processes generated by the QA system are as follows. The first step is text segmentation, which divides structured text into 1,200-character chunks. A 150-character overlap is applied between consecutive chunks to ensure contextual continuity. The overlap area prevents contextual discontinuity that can occur at chunk

boundaries, thereby improving the accuracy of entity and relationship extraction. The second step is to extract entities and relationships. The OpenAI GPT-4o model is used in each chunk to identify meaningful entities and extract relationships between them. Entities include various types, such as people, organizations, places, dates, and concepts, while relationships are expressed in sentences, such as affiliation, participation, location, and temporal relationships.

Third, in the embedding generation and storage step, the extracted entities and relationships are converted into 1,536-dimensional vectors using the OpenAI text-embedding-3-small model and stored in LanceDB. The vector embeddings are intended for fast retrieval based on semantic similarity. Fourth, in the graph construction and clustering stage, the NetworkX library is used to construct a knowledge graph by organizing entities as nodes and relationships as edges. The Leiden algorithm is then applied to group highly correlated entities into groups of up to 10, forming communities. A community represents a set of semantically closely related entities. Fifth, in the community analysis stage, the core semantic topics of each community are extracted and summarized using LLM. These community summaries are used for comprehensive query-answering (global search) of the entire dataset. Finally, in the knowledge database creation stage, the generated knowledge graph is saved as a Parquet file containing entity information, relationship data, community summaries, and text chunks. A GraphML file for knowledge graph visualization is also created.

Figure 4 shows a use case of a QA system created using QA Maker for the domain of the Dept of CSE at Seoul National University. This QA system provides an accurate and detailed response to the query "How do I apply for a teaching assistant position?", while the existing search system of the Department failed to address.

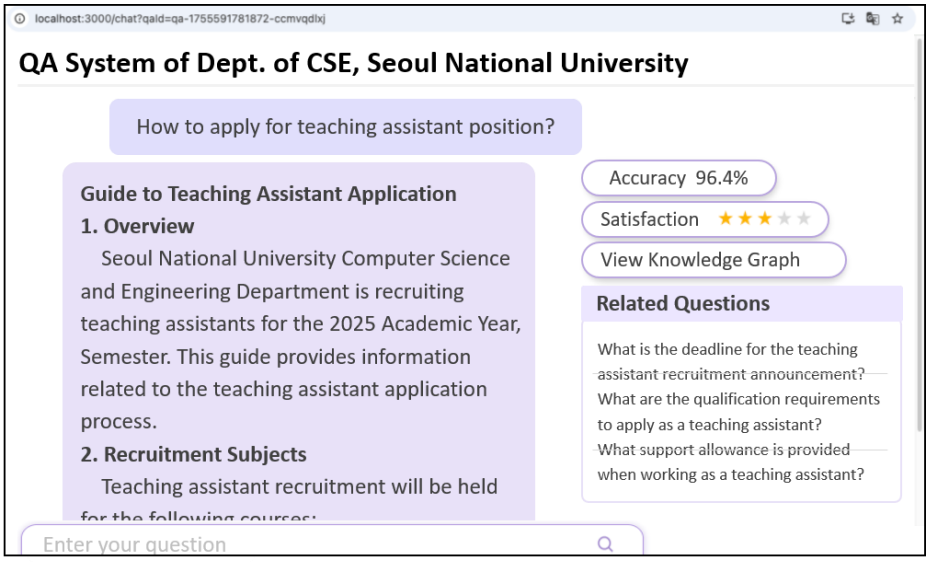


Figure 4. QA System generated by QA Maker for Dept. of CSE at SNU

4. Performance Validation

This section describes the results of experiments conducted to verify the performance of QA Maker.

4.1 Generation Time

To demonstrate how long it takes QA Maker to create a QA system, we measured the time it took to create a QA system for three randomly selected organizations, and the results are shown in Table 1. The computer

used in this experiment had an AMD Ryzen Threadripper PRO 5995WX 64-Cores × 12 CPU, 250 GB of memory, and was running Linux. The results of this experiment show that a QA system can be created within a few hours for a small domain such as KOS (Korea AI Software Industry Association), and within a few days for a large domain such as the entire Hansung University.

Table 1. Two sample QA systems generated by QA Makers

Domain	URLs	Documents	Web Crawling Time(minutes)	Structuring Time(hours)	Indexing Time(hours)	Total (hours)
Hansung University	11,937	4,417	63	7	31	39
Dept. of CSE, SNU	802	220	21	3	7	10
KOSA	142	12	20	0.21	4.5	5

4.2 Response Accuracy

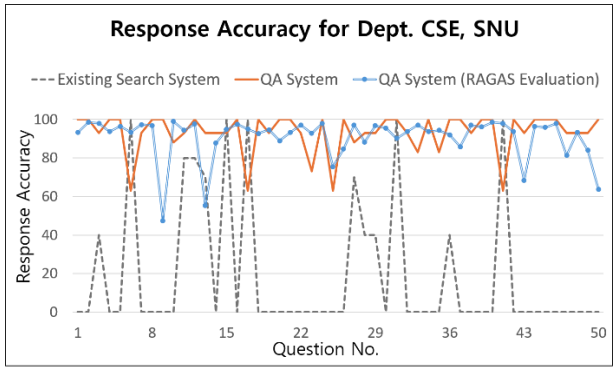
We selected two domains—the Department of CSE at Seoul National University and Hansung University—as test subjects and compared the response accuracy of existing search systems and the QA system generated by QA Maker for each domain. For this comparison, we randomly selected 50 user queries and the content that must be included in each query response. Using these, we conducted the following three evaluations

First, we formed a team of ten human evaluators and asked them to subjectively rate the accuracy of responses to 50 queries prepared for the existing search system and the QA system generated by QA Maker for each test domain, on a scale of 0 to 100. Furthermore, we had them rate the accuracy of responses to the same 50 queries using the RAGAS (RAG Assessment) framework, which is used to evaluate the quality of search results in information retrieval systems. The four metrics used for evaluation in the RAGAS framework are F (Faithfulness), R (Answer Relevancy), C (Context Recall), and P (Context Precision), which are all real values between 0 and 1, and the response accuracy, Accuracy is calculated as in the following equation (1) and is a real value between 0 and 100.

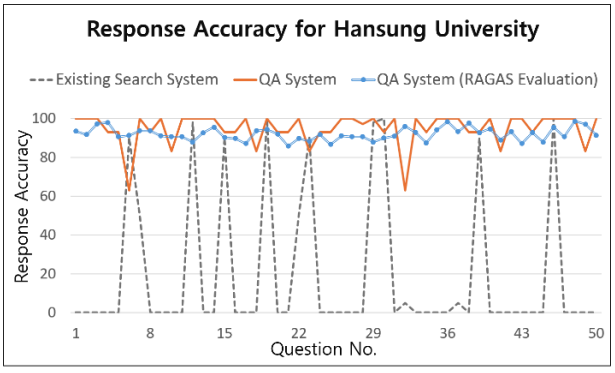
$$Accuracy = \frac{35 \cdot F + 30 \cdot R + 20 \cdot P + 15 \cdot C}{100} \quad (1)$$

Figure 5 shows the evaluation results for three response accuracies. Figure 5(a) compares the response accuracy values for CSE domain at the Seoul National University, while Figure 5(b) compares the response accuracy values for the Hansung University domain.

As a result of the experiment, the response accuracy of the existing search system, as rated by human evaluators for each test domain, was less than 20 points on average, while the response accuracy of the QA system was over 90 points on average. In addition, the response accuracy rated by the QA system itself using the RAGAS framework was over 90 points on average.



(a) Response Accuracy for Dept. of CSE, SNU



(b) Response Accuracy for Hansung University

Figure 5. Response Accuracy Comparison

5. Conclusion

This paper proposed and implemented QA Maker, a technology that automatically generates domain-specific QA(question-answering) systems using GraphRAG technology, and verifies its practicality. QA Maker automatically crawls a domain's URL, collecting all subdomain web pages and attached documents and building a knowledge graph database. This minimizes the cost and time required to build a QA system. Furthermore, a comparative evaluation of the response accuracy between a conventional search system and a QA system generated by Maker across two test domains revealed that the response accuracy of the conventional search system was less than 20%, while the response accuracy of the QA system was over 90%. In conclusion, the QA Maker proposed in this paper is considered a highly practical solution for establishing a domain-specific QA system.

Acknowledgement

This research was financially supported by the Hansung University

References

[1] M. Duka, M. Sikora, and A. Strzelecki, "From Web Catalogs to Google: A Retrospective Study of Web Search Engines Sustainable Development," Sustainability, Vol. 15, No. 8, 2023. DOI: https://doi.org/10.3390/su15086768

[2] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, Vol. 33, pp. 9459–9474, 2020.

[3] J. J. Pan, J. Wang, and G. Li, "Survey of Vector Database Management Systems," The VLDB Journal, Vol. 33, No. 5, pp. 1591–1615, 2024. DOI: https://doi.org/10.1007/s00778-024-00864-x

[4] Xu, X. Zhu, Y. Xie, Y. Liu, Y. Li, and W. Hu, "Knowledge Graph-Guided Retrieval Augmented Generation," Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL 2025), pp. 8912–8924, 2025. DOI: https://doi.org/10.18653/v1/2025.naacl-long.449

[5] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, "RAGAs: Automated Evaluation of Retrieval Augmented Generation," Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (EACL 2024), pp. 150–158, 2024. DOI: https://doi.org/10.18653/v1/2024.eacl-demo.16

[6] B. F. Green Jr., A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: An Automatic Question-Answerer," Proceedings of the Western Joint IRE-AIEE-ACM Computer Conference (IRE-AIEE-ACM '61), pp. 219–224, 1961. DOI: https://doi.org/10.1145/1460690.1460714

[7] L. Huang et al., "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions," ACM Transactions on Information Systems, Vol. 43, No. 2, pp. 1–55, 2025. DOI: https://doi.org/10.1145/3703155

[8] K. Shuster, S. Poff, M. Chen, D. Kiela, and J. Weston, "Retrieval Augmentation Reduces Hallucination in Conversation," in Findings of the Association for Computational Linguistics: EMNLP 2021, pp. 3784–3803, 2021. DOI: https://doi.org/10.18653/v1/2021.findings-emnlp.320

[9] X. Zhu, D. Yi, H. Jiang, C. Shi, and Y. Wang, "Knowledge Graphs: Opportunities and Challenges," Artificial Intelligence Review, Vol. 56, pp. 13071–13102, 2023. DOI: https://doi.org/10.1007/s10462-023-10465-9

[10] Ye-Won Seo, Yeon-Seo Bae, You-Jeong Jeong, Kitae Hwang, “Knowledge Graph of Smartphone Photos”, The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 24, No. 5, pp.203-209, Oct. 2024. DOI: https://doi.org/10.7236/JIIBC.2024.24.5.203

[11] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv preprint, arXiv:2404.16130, 2024. DOI: https://doi.org/10.48550/arXiv.2404.16130

[12] J. Ok, J. Soung, C. Park, and K. Hwang, “Implementation Details of EPUB Reader using GraphRAG,” International Journal of Internet, Broadcasting and Communication, Vol. 17, No. 2, pp. 223–231, May 2025. DOI: https://doi.org/10.7236/IJIBC.2025.17.2.223

[13] J. Gruber, "Markdown," Daring Fireball, 2004. https://daringfireball.net/projects/markdown/