AzureIQ-RAG: Hybrid RAG System for Automated Information Retrieval and Semantic Query Answering Using Azure AI

Ishtiaque Mahmood
Dept. of Computer Science and MIS
Oman College of Management and Technology, Oman
Barka, South AI Batinah, Oman
iabdulsattar@ocmt.edu.om

Mahammad Mastan
Dept. of Computer Science and MIS
Oman College of Management and Technology, Oman
Barka, South AI Batinah, Oman
mastan.mohammed@ocmt.edu.om

Zunysha Naveed
Department of Computer Science
University of Engineering and Technology, Taxila
Taxila, Pakistan
21-cs-75@students.uettaxila.edu.pk

Hamza Iftikhar
Department of Computer Science
University of Engineering and Technology, Taxila
Taxila, Pakistan
21-cs-45@students.uettaxila.edu.pk

Wakeel Ahmad
Department of Computer Science
University of Engineering and Technology, Taxila
Taxila, Pakistan
wakeel.ahmad@uettaxila.edu.pk

Syed M. Adnan
Department of Computer Science
University of Engineering and Technology, Taxila
Taxila, Pakistan
syed.adnan@uettaxila.edu.pk

2025 12th International Conference on Information Technology (ICIT) | 979-8-3315-0894-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICIT64950.2025.11049144

***Abstract*— Extracting relevant information from documents and websites is a time-consuming task that becomes increasingly complex as document length grows and websites expand. Manually retrieving precise information to meet user requirements from large documents and extensive websites is challenging and inefficient. In this context, Artificial Intelligence (AI) has emerged as a transformative technology, offering unparalleled capabilities for automating systems. AI has been widely utilized to streamline various processes, demonstrating its potential to simplify complex tasks, enhance accuracy, and reduce human effort. This research paper introduces "AzureIQ-RAG" Azure powered Intelligent Query and Hybrid Retrieval Augmented Generation (RAG) system with two primary features: (1) extracting and retrieving information from diverse document formats, including PDF, TXT, DOCX, etc. and enabling question-answering based on the provided document(s), (2) extracting content from website and enabling question-answering regarding the content of the website. The system automates the extraction process from both documents and websites. The extracted content is segmented, embedded using Azure OpenAI Embeddings, and indexed in Azure Cognitive Search for efficient retrieval and then a reranking mechanism is applied to improve precision and accuracy further. Then the final response is generated by passing the relevant content and user query to the Large Language Model (LLM) GPT -4 omni. Additionally, the system maintains separate session memory for both document and website searches, ensuring context retention for improved query handling and user experience. The document processing feature of the system has been tested on a dataset of over 950+ documents of various formats and achieved an accuracy of 94.77 %, and the website feature of the system is tested on 50+ different websites, achieving an accuracy of 75.22%.**

Index Terms*— *AI, Information Retrieval, Document Analysis, Semantic Search, Azure OpenAI Embeddings, Azure Cognitive Search, Automated Text Extraction, Multiformat Document Processing.

I. INTRODUCTION

Extracting the right information from any medium is crucial, as it ensures accurate and relevant information is retrieved while reducing noise and irrelevant data. This is especially important for decision-making, research, and efficient knowledge management.

Documentation is critical for organizations in all industries, whether in the form of handwritten or electronic records. In today's fast-paced world, where technology is rapidly evolving, most organizations store their documents electronically. These digital documents can be in a variety of formats, including PDF, XLS, CSV, TXT, and more.

As the size of a document grows—regardless of its formats, the complexity of analyzing and extracting useful information from documents based on specific requirements increases. Documents can contain a variety of information, such as employee records, organizational SOPs, product specifications, applications, and official letters.

Manually extracting useful information from documents is time-consuming and error prone. For example, if someone is assigned the task of summarizing a report but is unable to understand its context due to unfamiliarity or other factors, extracting key information becomes even more difficult.

Websites are created for different purposes, such as news websites, research papers, promotional websites, etc.

Like documents, as websites increase in terms of content and information, extracting useful insights based on user demands becomes difficult and challenging. Manually retrieving the information is inefficient as well as costly in terms of time. Retrieving information from any source comes with its advantages and disadvantages[1]. However, considering the length and complexity of the content, the manual process is highly costly in terms of time. In today's rapidly growing world, time is a critical resource, making automation a more efficient alternative.

Recent advances in AI, particularly Large Language Models (LLMs) [2] and RAG systems, have significantly enhanced the automation and increased the efficiency of complex tasks across various domains and industries [3-5]. Aligning LLM and RAG technologies can greatly enhance system capabilities, making them more beneficial across multiple industries. [6]. While manual Information retrieval (IR) remains an option, automating IR task from both sources: documents and websites, aligning RAG and LLM, presents significant challenges.

The use of RAG and LLM for IR introduces multiple challenges such as:

1. **Managing Lengthy Content**: As the length of the content increases, it becomes difficult for an LLM to process the entire document at once and accurately

979-8-3315-0894-4/25/$31.00 ©2025 IEEE

answer the user's query if the document is directly passed to it. To address this issue, content from websites or documents should be divided into smaller chunks. Then the most relevant chunk according to the user query must be retrieved and passed to the LLM for accurate response [7].

2. **Ambiguous User Questions:** Natural languages are inherently ambiguous, which can result in difficulty in understanding user queries and may lead to incorrect response generation. To overcome this problem, the natural language processing (NLP) model used should be able to understand ambiguous and poorly structured user queries. Additionally, if the user provides input with additional information about their requirements, it can help improve the accuracy of retrieval and response generation [8].
3. **Accurate Retrieval of Information:** The goal is to retrieve information that satisfies the user's query, and using keyword-based matching is not the right approach, as it only matches keywords irrespective of their context within the user query, resulting in incorrect information being retrieved. To overcome this problem, semantic search is used with vector embeddings, which understand the context of the user's question and retrieve information based on context [9].
4. **Handling Complex Queries:** Sometimes, user queries are complex enough to be handled as a single query. The information that should be retrieved for these types of queries may come from different parts of the documents. To handle this complexity, the user query should be broken down into sub-queries and treated separately as individual questions or queries. The results of all sub-queries are then combined at the end to provide a cumulative single result for the actual query the user has given as input.

To overcome these challenges, AI researchers and other professionals must work together. Furthermore, NLP models should be continuously improved to accurately interpret user input and produce correct and precise responses. In addition, embedding models should be optimized to improve content matching, resulting in more relevant and context-aware information retrieval.

This proposed research paper thoroughly investigates the use of an RAG-based system along with LLM capabilities to automate the information retrieval task, and allowing users to freely interact with the system and retrieve the most relevant information.

Following are the key contributions of our research:

1. **Session Management and Storage:** This system maintains the history of previous user chat which allows the user to complete the previous session, or it can use the content of that for future retrieval. All the past information is stored in the form of its sessions separately.
2. **Explicit Re-Ranking:** After retrieval of relevant chunks according to user query, those chunks are passed through an explicit re-ranking step to improve accuracy. In this step, the similarity score between the user query and each chunk is calculated, and the top *k* chunks with the highest similarity scores are selected and passed to the LLM.
3. **Handling Complex Queries with Sub-Query Decomposition**: When the user enters a complex query, it is first divided into sub-questions and handled separately for chunks retrieval and reranking. The selected chunks after re-ranking will be passed to LLM along with the complete user question.

This novel approach not only enhances IR efficiency but also saves time by eliminating the need for manual document processing in terms of IR. Additionally, it effectively handles complex queries regardless of the length of the content to process.

II. Literature Review

IR systems have evolved with advancements in technology, particularly in AI and LLMs. Over time, better methods and technologies have been discovered to make IR systems faster, more accurate, and efficient.

Early IR systems were based on keyword matching or statistical methods. Term Frequency-Inverse Document Frequency (TF-IDF) and BM25 were initially used for optimized keyword-based retrieval. [10]. However, these early methods lacked contextual understanding, leading to the development of semantic indexing, which captures latent semantic relationships in documents.

One such advancement is Sentence-BERT (SBERT), which modifies the original BERT model by employing Siamese and triplet network structures. This method generates semantically meaningful sentence embeddings, allowing for efficient document comparison using cosine similarity. Notably, SBERT significantly reduces processing time to be only 5 seconds while maintaining the accuracy of traditional BERT models. [11]. Similarly, Latent Semantic Analysis (LSA) improves search relevance by analyzing hidden structures within text, enhancing clustering and document ranking. [12]. However, LSA faces limitations in precisely mapping user queries to relevant content.

To address these challenges, deep learning-based approaches have become integral to modern IR systems. The Deep Structured Semantic Model (DSSM), for instance, projects queries and documents into a shared low-dimensional space, where the distance between them indicates semantic relevance [13].

A more recent innovation in IR is RAG, which bridges retrieval-based and generative approaches. Unlike purely parametric models, RAG first retrieves relevant information from an external knowledge base before passing it to a generative model to construct a response. This method improves the reliability and factual consistency of generated outputs. [14]. In the initial RAG framework, both retrieval and generation components were neural networks trained together on question-answering tasks. However, early studies highlighted the risk of the generative model disregarding retrieved content when the system was not optimally tuned.

A notable refinement of RAG involves a modular design, where the retrieval and generation components are developed independently. This approach enables greater flexibility in combining different retrieval techniques with LLMs. Various enhancements have further refined RAG's performance, including query rewriting for improved search results. [15],

metadata integration for contextual awareness, and leveraging structured data sources such as knowledge bases and tabular datasets [16, 17]. Advanced retrieval techniques, such as passage re-ranking and iterative retrieval for multi-hop reasoning, have also been introduced to improve accuracy. [18, 19].

Another area where LLMs have shown promise in IR is in generating high-quality search inputs. Recent work on automated testing and fuzzing techniques has demonstrated how LLMs can be leveraged to improve data retrieval, for instance, ChatFuzz integrates generative AI to create structured variations of seed inputs, which are assessed and incorporated into the AFL++ greybox fuzzer [20].

Similarly, the SeedMind framework synthesizes test case generators rather than direct inputs, refining them iteratively through feedback loops to enhance seed quality. Experiments conducted on 166 programs and 674 harnesses from OSS-Fuzz and the MAGMA benchmark demonstrated that SeedMind outperforms state-of-the-art LLM-based solutions, such as OSSFuzz-AI, achieving up to 27.5% higher code coverage. [21].

III. Proposed Methodology

The system consists of two features:

1. Extracting content from various document formats, including PDF, TXT, DOCX, etc., and enabling question-answering based on the provided documents.
2. Extracting content from a website and enabling question-answering regarding the content of the website.

In both features, the content extraction process differs. For the implementation of both features, we have leveraged Microsoft Azure's cloud platform for seamless model access, ensuring reliability and secure integration of AI services.

The process begins by allowing the user to either enter a website link to extract data, upload multiple documents in various formats such as PDF, XLSX, and DOCX to extract data from them, or continue a previous session if desired. The process of extracting data from documents differs from that of extracting data from a website.

A. Data Extraction from Documents

Documents exist in various formats, such as PDF, DOCX, XLSX, and many more. User can upload document(s) in any format. Once document(s) is successfully uploaded, its content is extracted using specialized Python libraries and explicit functions designed for this purpose. If the document contains any images, OCR technology is used to extract the textual content from them.

B. Data Extraction from Website

If the user enters any website URL for analysis, the system then utilizes Python libraries such as BeautifulSoup to scrape the content available on the webpage.

Once the data is extracted, according to the user question it is then processed further.

The extracted content is divided into chunks, each with a size of 500.

Let C represent the extracted content from document(s) or website. The chucking process will be formulated as:

$$C=\{c_1,c_2, \dots ,c_n\} \text{ where } |c_i| \leq 500$$

Where:

- c_i is an individual chunk.
- $|c_i|$ represents the number of the words in each chunk.

Embeddings $E(c_i)$ for each chunk are calculated once the chunking process is done by using Embedding-3 Large model.

$$E(c_i) = f(c_i), \; E(c_i) \in R^d$$

Where:

- **f** is the function of embedding
- **d** is the dimension of embedding

This model understands context better than previous versions and is highly effective with minimal labeled data, making it ideal for zero-shot or few-shot learning.

Moving forward, both the content and embeddings are indexed in the Azure AI Search Service.

In Azure AI Search, an index is created with three fields:

1. **id** (of type emd.string)
2. **content** (of type emd.string)
3. **embeddings** (of type emd. collection(single))

The extracted content and corresponding embeddings are then stored in these fields, and a unique ID is assigned to each chunk by hashing the chunk.

In the case of multiple documents being provided as input, chunks are created, embedded, and stored for all documents simultaneously.

Once the data is indexed in the Azure AI Search Service, the user can ask any questions they wish. The user may ask multiple questions related to the same document, different documents, or the same website, depending on whether they are currently dealing with document(s) or a website.

If the user asks multiple questions, then a list of questions is created, and each question is treated separately as a sub-question.

Once the user has entered their question(s), Azure Search Service retrieves the relevant chunks.

The service first generates embeddings (Embeddings are numerical vector representations of data such as words, sentences, images, or documents in a continuous vector space) for the user question using the same model that was used for creating embeddings of the chunks. Then, **semantic search** is performed by **Azure AI Search Service** to retrieve the most relevant chunks. If multiple questions are provided, the service retrieves relevant chunks separately for each one. In total, the top 10 chunks are retrieved for every question depending upon the similarity between the embedding of the user question and the embedding of the chunk.

Once the chunks are retrieved by Azure Search Service for the user's question(s), any duplicate chunks are removed, which may occur in the case of multiple questions. After removing duplicate chunks, external reranking is performed to ensure a more precise and accurate response generation.

In external reranking, cosine similarity is used to identify the chunks that are most relevant to the user's query. The similarity score is calculated between the user question(s) and the retrieved chunks. In the case of multiple questions, the similarity score is determined for each question against every retrieved chunk. Each chunk is then assigned the highest similarity score it receives from different sub-questions. After assigning scores to each chunk, the top k chunks with the highest scores are selected as the final chunks. In our system, the value of k is set to 7.

Cosine Similarity measures the similarity between two vectors in multi-dimensional space by calculating the cosine of the angle between them. The closer the cosine similarity is to **1**, the more similar the vectors are.

Given a user query Q, its embedding is computed as:

$$\mathbf{E(c_i) = f(c_i),\ \ E(c_i) \in R^d}$$

The relevance of each chunk is computed using **cosine similarity**:

$$\textit{Cosine Similarity} = \frac{E(Q).E(c_i)}{|E(Q)| \times |E(c_i)|} \quad (1)$$

Where:

- $E(Q).E(c_i)$ is the dot product.
- $|E(Q)|$ and $|E(c_i)|$ are the magnitudes of the vectors.
- The top **7** chunks with the highest similarity scores are retrieved.

Moving forward, the final chunks along with the user question(s) are passed to the GPT-4 omni to generate the response.

External reranking is effective in terms of resource utilization and accuracy, because the shorter the prompt will be to the LLM the smaller the number of tokens is given as input, and the LLM will be able to understand the context better because there will be no irrelevant information concerning the user question(s).

If the user does not start a new session and chooses to continue a previous session, the stored data for that session is retrieved from the Chroma database, where sessions are saved individually. The system allows us to set a time limit for retaining sessions; in our case, sessions are stored for 10 days.

Once the user selects a previous session, the corresponding data is loaded, enabling them to seamlessly continue asking questions without restarting the process.

Algorithm

Input: UserInput (documents D or website URL U),
Queries Q = {q_1, q_2, ..., q_n}

Output: Responses R = {r_1, r_2, ..., r_n}

if UserInput contains D then:
  T ← extract_content(D) // Uses OCR for images
elif UserInput contains U then:
  T ← BeautifulSoup.scrape(U) // Web scraping
C ← chunk_text (T, size=500)
E ← Embedding-3-Large(C) // Generate embeddings
index ← AzureAISearch.create_index (fields=[id, content, embeddings])
AzureAISearch.store(index, hash(C), C, E)
retrieved_chunks ← []
for each q in Q **do**:

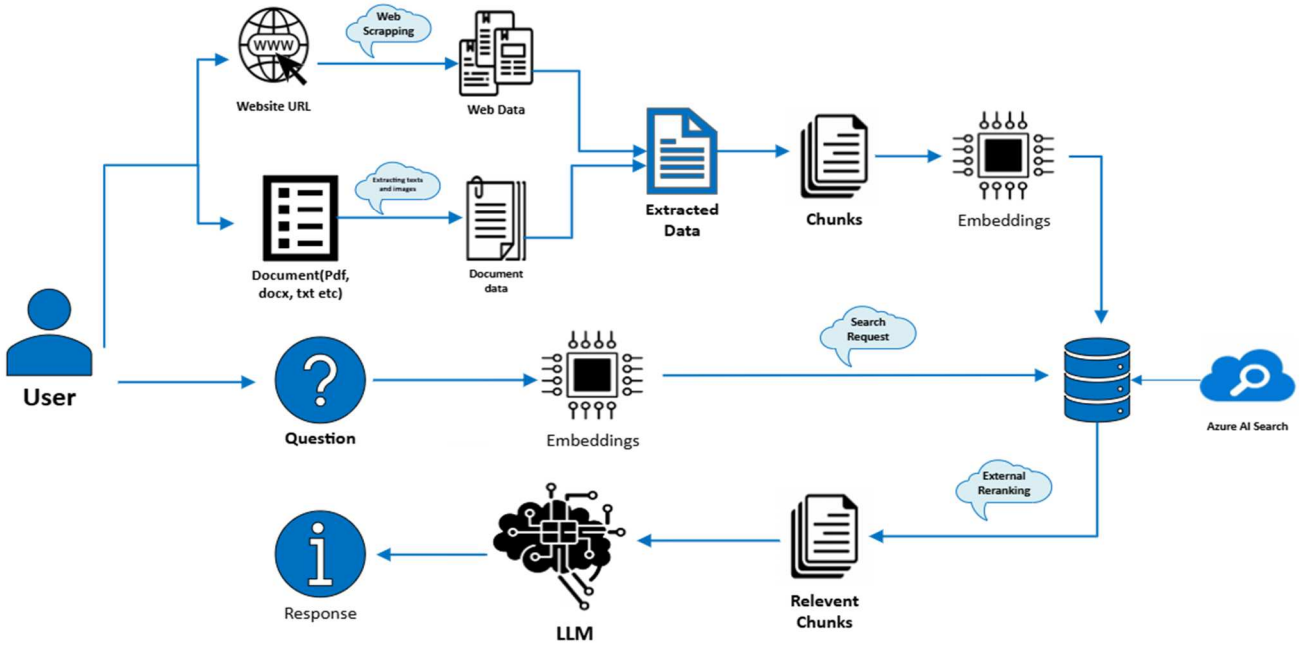

Figure 1 Overview of proposed AzureIQ-RAG Hybrid RAG System for Intelligent Retrieval and Semantic query answering using Azure AI, basic workflow diagram.

```
    chunks ← AzureAISearch.semantic_search(q, k=10)
    retrieved_chunks.extend(chunks)
uniq_chunks ← remove_duplicates(retrieved_chunks)
reranked_chunks ← []
Q_embeddings← []
for each q in Q do:
    Q_embeddings.append(Embedding-3-Large(q))
for each c in uniq_chunks do:
    scores ← [cosine_sim(q_emb, c_emb) for q_emb in Q_embeddings]
    max_score ← max(scores)
    reranked_chunks.append((c, max_score))
top_chunks ← sort(reranked_chunks).take(k=7)
R ← GPT-4-Omni(Q, top_chunks)
Return R                                   // Final Response
```

IV. EXPERIMENTATIONS AND RESULTS

A. Dataset Used

Our dataset consists of two types of documents, divided based on chunk size: one set with a chunk size of 300 tokens and another with a chunk size of 500 tokens. It spans multiple domains and includes multi-format documents such as DOCX, XLSX, TXT, CSV, and PDF, covering reports, policies, letters, etc. Additionally, for our web-based retrieval system, we have generated a dataset of 50 website URLs and then the entire system is evaluated on user questions related to documents and websites.

Table 1 Details of the Dataset used for experimentation. Chunk Size is given in tokens.

Chunk Size (Tokens)	*No. of Documents*	*Avg. Document size (Tokens)*
300	580	865
200	375	1032

Web-based retrieval is evaluated using a total of 230 questions related to web URLs, while document-based retrieval is tested with 2,067 questions generated for multiple documents.

B. Results

In this study, we evaluated multiple retrieval models to determine their effectiveness in retrieving relevant document chunks. Our analysis focuses on using different chunk sizes and retrieval techniques. For document embedding, we utilized Microsoft embedding-3-large to generate semantic representations. To evaluate retrieval performance, we employed three different retrieval models: Azure AI Search, BM25, and MiniLM-L12-v2. Retrieval accuracy was assessed using the original questions, computing the percentage of cases where the correct chunk appeared in the Top 3 and Top 5 results.

Different chunk sizes are used to evaluate the performance of models. The evaluation of models for different chunk sizes based on retrieval accuracy is given below in table 2.

Table 2- Top 3 and Top 5 accuracies of different models for retrieval. The best results are highlighted.

	Size = 300		*Size=500*	
Models	***Top 3***	***Top 5***	***Top 3***	***Top 5***
MiniLM-L12-v2	0.24	0.41	0.17	0.23
BM25	0.32	0.49	0.31	0.50
Azure AI Search	**0.41**	**0.57**	**0.42**	**0.59**

These results indicate that Azure AI Search consistently outperformed BM25 and MiniLM-L12-v2 across both chunk sizes. The BM25 model, which relies on term frequency-based ranking, showed competitive performance, particularly in the Top 5 retrievals. This suggests that a moderate chunk size enhances retrieval effectiveness by preserving more contextual information. In contrast, MiniLM-L12-v2, despite being a neural model, had lower accuracy, particularly when the chunk size increased to 500 tokens due to difficulty in encoding larger segments effectively.

The results highlight that traditional lexical retrieval (BM25) remains strong while hybrid approaches (Azure AI Search) leveraging both keyword-based and semantic search offer the best performance.

To assess the quality of answers produced by GPT-4o, we employed multiple evaluation metrics. First, the F1 score is measured using recall and precision, which captures the overlap of words between the generated and reference answers. This helped evaluate lexical accuracy. Next, cosine similarity was used to compare the embeddings of the predicted and reference answers, ensuring that even differently phrased responses were assessed for semantic closeness. Additionally, GPT-4o itself was used to rate the generated answers on a scale of 0 to 100 based on correctness and relevance.

Table 3- F1 Score, Cosine Similarity and LLM score of GPT-4o using different number of retrieval documents(K).

	K=5			**K=7**		
Model	**F1**	**Cos.**	**LLM**	**F1**	**Cos.**	**LLM**
GPT-4o	48.71	94.2	56.4	49.2	94.77	48.7

Here “K” is the number of retrieved documents in the form of chunks.

These results indicate that retrieving more document chunks does not always improve the quality of GPT-4o's responses. While the semantic similarity score remains consistently high (~94%), the F1 score remains below 50, highlighting a disparity between lexical accuracy and semantic relevance. Additionally, the variation in LLM-based scores suggests that GPT-4o may struggle with over-retrieval, as excessive

context can introduce irrelevant information, potentially reducing answer clarity.

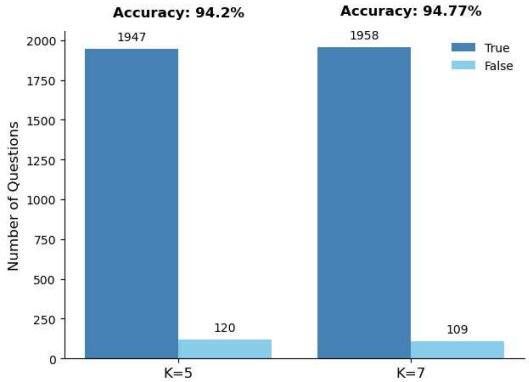

Figure 2 Accuracy of AzureIQ-RAG upon retrieval of different number of chunks.

We evaluated our web-based RAG system using a set of 230 questions. Out of these, the system provided correct responses for 173, resulting in an accuracy of 75.21%. This evaluation helps in understanding the system's ability to retrieve relevant information and generate precise answers. These results are also calculated upon the basis of cosine similarity score between predicted and actual answer.

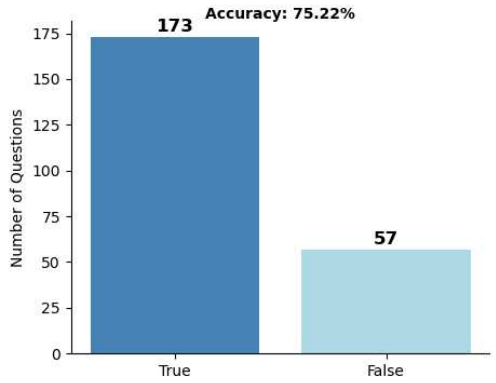

Figure 3 Accuracy of AzureIQ-RAG for Web-based Information Retrieval

Our findings demonstrate that hybrid retrieval (Azure AI Search) leads to superior performance compared to traditional lexical methods (BM25) and neural embedding-based models (MiniLM-L12-v2). However, key challenges remain:

- Chunk size optimization is crucial, as retrieval effectiveness fluctuates with increasing token length.
- Semantic vs. Lexical trade-offs: GPT-4o-generated answers exhibit strong semantic consistency but lack precise word-level accuracy, highlighting potential limitations in F1-based evaluations.

Over-retrieval risks: Increasing K beyond an optimal threshold may introduce distracting context, negatively impacting LLM response correctness.

V. Conclusions

Our system effectively extracts and retrieves information from both structured documents and unstructured web content using advanced AI techniques. By leveraging Microsoft Azure, RAG, and GPT-4 Omni, it ensures scalable, precise, and efficient content processing. The integration of external re-ranking and semantic search further enhances retrieval accuracy while optimizing computational efficiency. Supporting a wide range of document formats, including PDF, DOCX, CSV, and TXT, along with dynamic web scraping, the system enables comprehensive data extraction. The chunking strategy, embedding via the Embedding-3 Large model, and indexing with Azure AI Search enhance retrieval speed and contextual understanding. Experimental results show significant retrieval accuracies of 94.77% and 75.22% for document-based retrieval and web-based retrieval, respectively. The explicit re-ranking used after extracting data from Azure AI Search has played a crucial role in achieving these high accuracies. It also benefits resource utilization by passing the most relevant information related to the user query to GPT-4 Omni. The system is evaluated on different chunk sizes, and the performance evaluation shows that a chunk size of 500 is the most optimal one. It ensures that no chunk contains an excessive amount of data, making it a well-balanced choice. Additionally, the system has been evaluated across multiple domains and diverse datasets, proving its adaptability and robustness. Its ability to retrieve and process both document-based and web-based data makes it a valuable solution for enterprise knowledge management, legal document analysis, policy extraction, and business intelligence. In summary, this AI-powered retrieval system successfully integrates state-of-the-art NLP techniques, transformer-based embeddings, and intelligent ranking mechanisms to deliver high-performance information retrieval.

VI. Acknowledgment

This research and publication were funded by the project titled "Enhancing HR and Payroll Systems with Advanced Conversational AI: A Strategic Initiative in Oman's Digital Transformation Landscape", supported by the Ministry of Higher Education, Research and Innovation, Oman (Project ID: BFP/RGP/ICT/24/454).
The authors express their gratitude for this financial support.

A. Statement and Declaration

- ***Competing Interests***

There are no conflicts of interest to declare.

- **Author's contribution statement**

All authors contributed equally to this research and the writing of the paper. They were involved in conceptualizing the study, designing the experiments, analyzing the data, and drafting and revising the paper.

VII. References

1. Karthigaikumar, P., et al. *Comparing Automated vs. Manual Approaches for Information Retrieval*. in *2023 2nd International Conference on Futuristic Technologies (INCOFT)*. 2023. IEEE.

2. Naveed, H., et al., *A comprehensive overview of large language models.* arXiv preprint arXiv:2307.06435, 2023.
3. Gebreab, S.A., et al. *LLM-based framework for administrative task automation in healthcare*. in *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*. 2024. IEEE.
4. Lu, Y., et al. *A Retrieval-Augmented Generation Framework for Electric Power Industry Question Answering*. in *Proceedings of the 2024 2nd International Conference on Electronics, Computers and Communication Technology*. 2024.
5. Álvaro, J.A.H. and J.G. Barreda, *An advanced retrieval-augmented generation system for manufacturing quality control.* Advanced Engineering Informatics, 2025. **64**: p. 103007.
6. Fan, W., et al. *A survey on rag meeting llms: Towards retrieval-augmented large language models*. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024.
7. Abney, S.P., *Parsing by chunks.* Principle-based parsing: Computation and Psycholinguistics, 1992: p. 257-278.
8. Gao, T., et al. *Datatone: Managing ambiguity in natural language interfaces for data visualization*. in *Proceedings of the 28th annual acm symposium on user interface software & technology*. 2015.
9. Guha, R., R. McCool, and E. Miller. *Semantic search*. in *Proceedings of the 12th international conference on World Wide Web*. 2003.
10. Robertson, S.E. and S. Walker. *Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval*. in *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*. 1994. Springer.
11. Reimers, N. and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks.* arXiv preprint arXiv:1908.10084, 2019.
12. Evangelopoulos, N.E., *Latent semantic analysis.* Wiley Interdisciplinary Reviews: Cognitive Science, 2013. **4**(6): p. 683-692.
13. Huang, P.-S., et al. *Learning deep structured semantic models for web search using clickthrough data*. in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013.
14. Lewis, P., et al., *Retrieval-augmented generation for knowledge-intensive nlp tasks.* Advances in neural information processing systems, 2020. **33**: p. 9459-9474.
15. Ma, X., et al. *Query rewriting in retrieval-augmented large language models*. in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023.
16. Wang, X., et al., *Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases.* arXiv preprint arXiv:2308.11761, 2023.
17. Chen, W., et al., *Hybridqa: A dataset of multi-hop question answering over tabular and textual data.* arXiv preprint arXiv:2004.07347, 2020.
18. Zhuang, S., et al., *Open-source large language models are strong zero-shot query likelihood models for document ranking.* arXiv preprint arXiv:2310.13243, 2023.
19. Shao, Z., et al., *Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy.* arXiv preprint arXiv:2305.15294, 2023.
20. Fioraldi, A., et al. *{AFL++}: Combining incremental steps of fuzzing research*. in *14th USENIX workshop on offensive technologies (WOOT 20)*. 2020.
21. Shi, W., et al., *Harnessing Large Language Models for Seed Generation in Greybox Fuzzing.* arXiv preprint arXiv:2411.18143, 2024.
22. Torrey, L. and J. Shavlik, *Transfer learning*, in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. 2010, IGI global. p. 242-264.