

SeekMindRAG: Cost-Effective Hybrid Retrieval-Augmented Generation System on Consumer-Grade GPUs

Group Number: 12

Group Members(5):

Yuhe Ouyang (72403188)

Jie Wang (72401957)

Lijun Xu (72406191)

Xianjun Zhang (72406025)

Longkang Wang (72404124)

City University of Hong Kong (Dongguan)

April 2025

1. Introduction

The field of Natural Language Processing (NLP) has witnessed remarkable progress in recent years, particularly with the advent of Large Language Models (LLMs) and their powerful text generation capabilities. However, purely generative models often suffer from issues such as hallucination—generating factually incorrect or nonsensical information—and difficulty in incorporating timely or domain-specific knowledge not explicitly present in their training data.

To overcome these challenges, Retrieval-Augmented Generation (RAG) systems have emerged as a prominent paradigm in current NLP research and applications. RAG systems combine information retrieval mechanisms with generative models. Before generating a response, they retrieve relevant snippets or information from one or more external knowledge sources (such as document repositories, databases, or web pages) based on the user's query. Subsequently, the generative model synthesizes the final answer conditioned on the retrieved information. This approach effectively merges the model's "parameterized knowledge" (learned during training) with "non-parameterized knowledge" (from external knowledge bases), significantly enhancing the accuracy, relevance, and interpretability of the generated content, enabling them to provide knowledge-grounded responses to specific questions.

Despite the profound potential demonstrated by RAG systems in improving generation quality, their widespread deployment and application face significant challenges, particularly concerning hardware resources. Building efficient RAG systems typically involves processing large-scale document indexing and vector databases, loading and running generative models with massive parameter counts, and executing complex retrieval and inference computations. Consequently, traditional RAG deployment solutions are highly demanding in terms of computational resources, often necessitating reliance on high-performance, server-grade Graphics Processing Units (GPUs), such as NVIDIA A100, H100, among others. While these professional-grade hardware units offer exceptional computational power and large memory capacity, their acquisition and maintenance costs are prohibitive, and their supply can be limited. This constitutes a major technical and economic barrier for smaller developers, research institutions, startups, or individual users with

limited resources, thereby hindering the democratization and widespread adoption of RAG technology.

It is within this context that the present study/project is undertaken, aiming to explore the feasibility of implementing high-performance RAG systems on more accessible hardware platforms. We identify that there is a current lack of open-source RAG solutions that are deeply optimized and integrated specifically for consumer-grade hardware environments, particularly mainstream desktop graphics cards. To address this gap, we propose the **SeekMindRAG** project. SeekMindRAG's core objective is to democratize RAG capabilities by judiciously integrating and optimizing a suite of efficient open-source components and lightweight but performance-optimized open-source large language models (such as the DeepSeek series models optimized for specific needs). This approach is designed to enable stable and efficient operation on ubiquitous consumer-grade GPUs, such as the NVIDIA RTX 4060 equipped with 8GB of VRAM. We selected this type of hardware as our target platform considering its high penetration rate and affordability among a broad user base.

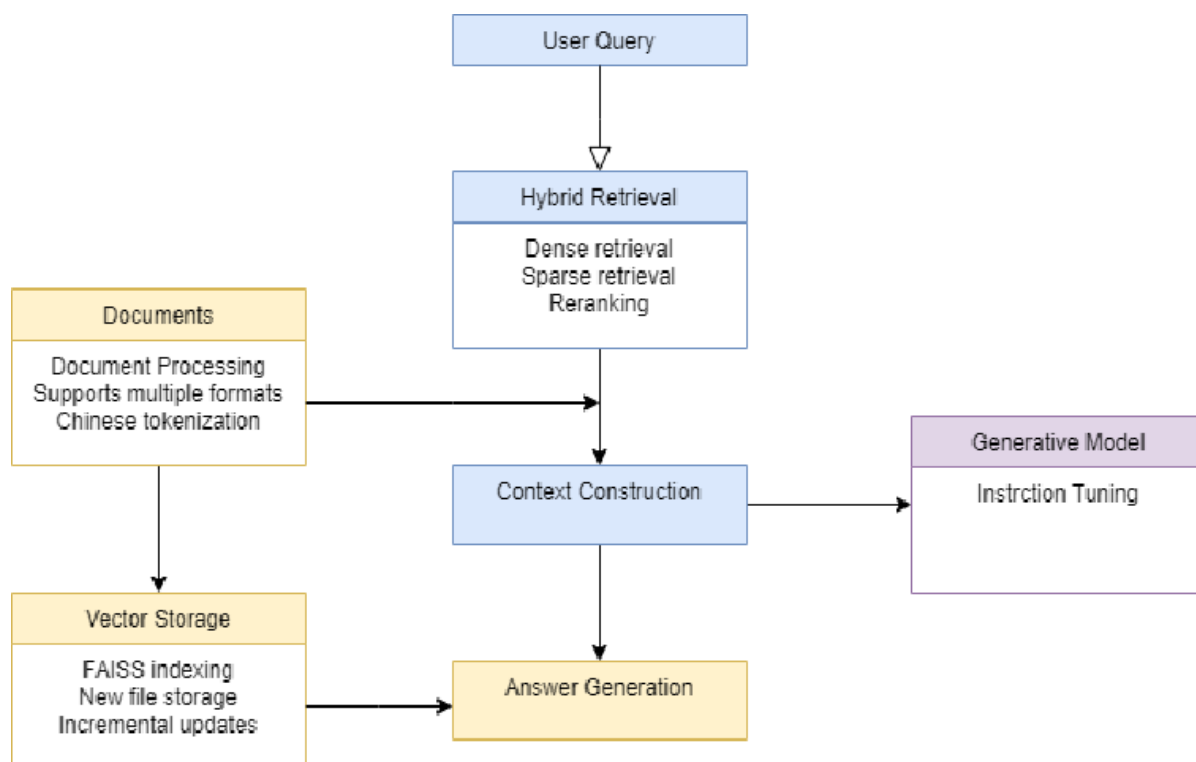
In terms of technical implementation, the SeekMindRAG project supports flexible ingestion and processing of multi-format documents, builds knowledge bases that support hybrid retrieval (including dense retrieval based on vector embeddings and sparse retrieval based on keywords/statistical features), and places a strong emphasis on optimizing inference efficiency to achieve fast response times under limited hardware resources. Ultimately, this project aims to provide an effective and scalable Chinese Q&A system solution based on consumer-grade hardware, offering support to more developers and users who wish to leverage RAG technology but are constrained by hardware limitations.

2. Related Work

Recent RAG implementations often prioritize performance at the expense of accessibility. Projects like LangChain and Haystack provide flexible pipelines but assume access to powerful compute resources. Other LLMs such as DeepSeek, Qwen, and Mistral have recently pushed toward lower resource consumption via quantization and distillation. Meanwhile, vector search engines like FAISS and hybrid retrieval frameworks combining BM25 and semantic embeddings have shown promise in boosting retrieval relevance. SeekMindRAG unifies these advancements into a streamlined, cost-effective stack.

3. Methodology

3.1 System Architecture



3.2 Architecture Overview

SeekMindRAG consists of four major functional modules:

- **Document Processor:** Ingests and chunks `.pdf`, `.docx`, `.txt`, `.md`, `.csv`, `.json`, and `.yaml` files using sliding window and sentence-aware strategies. It uses HanLP for Chinese tokenization with fallback strategies for robustness.
- **Vectorized Storage:** Uses FAISS for indexing with support for FlatL2, FlatIP, and HNSW strategies. Embedding generation is done using all-MiniLM-L6-v2 (FP16).
- **Retriever:** Combines FAISS dense retrieval and BM25 sparse retrieval, followed by reranking via cross-encoder (`ms-marco-MiniLM-L6-v2`).
- **Answer Generator:** Supports models like DeepSeek-R1-Distill-Qwen (7B, 1.5B) with 4-bit quantization. Prompt injection ensures controllable answer generation.
- **User-Friendly Chat Interface:** enables intuitive multi-turn conversations by managing chat history, displaying generated answers with transparent source references, and allowing selection of the underlying generation model.

3.3 Details of functional modules

3.3.1 Document Processor

This component is responsible for ingesting raw documents, processing them, and preparing them for indexing by breaking them into smaller, manageable chunks.

Key Features and Mechanisms:

1. Supported Formats:

- Processes documents in multiple common formats: `.pdf`, `.docx`, `.txt`, `.md`, `.csv`, `.json`, `.yaml`.

2. Chunking Strategies:

- Offers flexible chunking based on `chunk_size` and `overlap` parameters.

- Sliding Window:
 - Uses a token buffer to process text.
 - Reads text paragraph by paragraph.
 - Tokenizes the text and adds tokens to the buffer.
 - Creates a new chunk when the buffer size reaches or exceeds the specified `chunk_size`.
- Smart Sentence Breaking:
 - Avoids simple fixed-length truncation.
 - Identifies natural sentence end positions.
 - Finds the sentence end position closest to the desired `chunk_size` using an internal method (`_find_next_sentence_end`).
 - Recognizes common sentence end symbols like `.`, `!`, and `?`.

3. Tokenizer:

- Utilizes **HanLP**, a high-precision tokenizer specifically designed for Chinese text.

4. Overlapping Handling:

- Manages overlapping content between consecutive chunks.
- Retains content from the end of the previous chunk (equivalent to the `overlap` size) at the beginning of the next chunk.
- **Purpose:** This overlap mechanism helps maintain semantic coherence across adjacent chunks, which is crucial for improving retrieval accuracy during the RAG process.

5. Fault Tolerance:

- Includes mechanisms to ensure robust processing:
 - **Tokenizer Downgrade:** Automatically switches to a less complex or character-level segmentation if high-precision tokenization fails, ensuring stability.
 - **Indexing Error Handling:** Manages potential issues during chunking or indexing through intermediate splits to prevent failures.
 - **Empty Chunk Prevention:** Checks for and avoids generating chunks with empty content buffers.

3.3.2 Vectorized Storage

The Vectorized Storage component is responsible for converting processed text chunks into high-dimensional vectors (i.e., embeddings) and storing these vectors efficiently to enable fast similarity search. This is a core part of implementing the semantic retrieval functionality. SeekMindRAG's vectorized storage design focuses on efficiency and accessibility to meet the requirements for running on consumer-grade hardware.

Key Components and Characteristics:

1. Embedding Model:

- **Model:** `all-MiniLM-L6-v2`
- **Role:** This model is used to encode the text content of document chunks into fixed-length dense vectors (384 dimensions in this case). These vectors capture the semantic information of the text chunks.
- Optimization Characteristics:

- **FP16 Quantization:** Model weights and computations use half-precision floating-point numbers (FP16), which significantly reduces model size and memory footprint compared to standard FP32, and also speeds up inference.
- **Extremely Fast:** `all-MiniLM-L6-v2` is a lightweight model specifically designed for efficient encoding, capable of quickly generating embeddings on lower-configuration hardware.
- **Low Resources:** The model has a small footprint and low VRAM or RAM requirements, making it highly suitable for running on consumer-grade GPUs with 8GB VRAM or other resource-constrained environments. These characteristics align strongly with the project's goal of low hardware requirements.

2. Vector Index & Retrieval:

- **Core Technology: FAISS (Facebook AI Similarity Search)**
- **Role:** FAISS is a library developed by Facebook AI specifically for efficiently handling similarity search on large-scale vector data. It can quickly build indexes and find the nearest or approximate nearest neighbors for a given query vector.
- Supported Index Methods:

FAISS provides various index structures, and SeekMindRAG utilizes several of them:

- `IndexFlatL2` (Euclidean Distance): The simplest index type, performing exact L2 distance calculation, suitable for small datasets or scenarios requiring precise matching.
- `IndexFlatIP` (Inner Product Similarity): Computes the dot product of vectors. If vectors are normalized (often done for cosine similarity), this is equivalent to cosine similarity. Also suitable for small scale or precise scenarios.
- `IndexHNSWFlat` (HNSW Approximate Search): Recommended for approximate nearest neighbor search on large-scale datasets. HNSW (Hierarchical Navigable Small Worlds) is a graph-based indexing method that provides a good balance between search speed and accuracy, making it the preferred choice for processing a large number of document embeddings.
- **GPU Acceleration Support:** FAISS natively supports GPU acceleration, allowing indexes to be migrated to the GPU (`faiss.index_cpu_to_gpu`) for vector search, significantly boosting retrieval speed, especially when dealing with large indexes. This is crucial for leveraging the computational power of consumer-grade GPUs.
- **Dynamic Index Update:** Supports incrementally adding new documents (i.e., new embedding vectors) to the existing vector database. This means that when new information needs to be added to the knowledge base, there's no need to rebuild the entire index from scratch, improving system real-time performance and maintainability.

3. Local Secure Storage:

- **Method:** The vector index and associated metadata are stored in the local file system.
- **Advantages:** Simple deployment, no dependency on external database services. Data is stored in local files, providing a high level of security assuming the local machine environment is secure.

By combining an efficient embedding model with the powerful FAISS library and leveraging its GPU acceleration and dynamic update capabilities, SeekMindRAG's Vectorized Storage component can achieve fast, scalable knowledge retrieval on consumer-grade hardware, providing robust support for subsequent text generation.

3.3.3 Retriever

The Retriever component is responsible for finding the most relevant document chunks or passages from the knowledge base in response to a user's query. SeekMindRAG employs a hybrid approach to enhance retrieval performance.

Key Features and Mechanisms:

1. Hybrid Retrieval:

- Combines the strengths of both semantic (dense) and keyword (sparse) retrieval methods to improve recall and relevance.
- FAISS Dense Semantic Retrieval:
 - Uses deep learning-based embeddings to capture the semantic meaning of text.
 - Converts both documents (or document chunks) and the user query into dense semantic vectors.
 - Calculates the similarity between the query vector and document vectors (typically using FAISS indices) to find semantically relevant chunks.
 - Leverages the `a11-MiniLM-L6-v2` model for the vectorization process.
- BM25 Sparse Keyword Retrieval:
 - A traditional, statistics-based method that focuses on term frequency and inverse document frequency.
 - Performs exact keyword matching.
 - Uses `HanLP` for high-precision Chinese word segmentation before applying the BM25 algorithm.
 - Computes relevance scores based on the statistical occurrence and distribution of keywords.
 - Particularly effective at capturing results that contain the exact terms from the query, complementing the semantic approach.

2. Deduplication Optimization:

- A processing step applied to the retrieved results.
- Specifically designed to clean and filter out duplicate Chinese sentences, ensuring that the final set of retrieved information is concise and free of redundant content.

3. Rerank Mechanism:

- A post-retrieval step to refine the ranking of the initial results from the hybrid search.
- Separately retrieves results from both the dense (FAISS) and sparse (BM25) methods.
- Merges the results from both methods and performs deduplication on the combined list.
- Uses a **Cross-Encoder model** (`cross-encoder/ms-marco-MiniLM-L6-v2`) to perform a more precise, pairwise scoring of the merged results. This model considers the interaction between the query and each retrieved document chunk to provide a refined relevance score.
- Returns the final list of document chunks sorted according to the re-ranking scores.

3.3.4 Answer Generator

This module takes the original user query and the relevant document chunks retrieved by the Retriever component, and uses a Large Language Model (LLM) to synthesize a coherent and knowledge-grounded answer.

Key Features and Mechanisms:

1. Optional Generation Models:

- The module supports using different LLMs, offering flexibility based on desired performance and resource availability.
- **DeepSeek-R1-Distill-Qwen-7B:**
 - A larger model option.
 - Optimized for 4-bit local inference, allowing it to run on hardware with limited VRAM (like 8GB GPUs).
 - Provides high accuracy in generating answers.
- **DeepSeek-R1-Distill-Qwen-1.5B:**
 - A smaller, more lightweight model option.
 - Can be packaged and served via **Ollama**, potentially simplifying deployment and enabling faster response times.

2. Prompt Engineering:

- Dynamically constructs the input prompt for the chosen LLM.
- **Automatically injects the retrieved content** alongside the user's original query into the prompt.
- Designs the prompt to help **control the answer style and length** of the generated response, guiding the LLM's output format and verbosity.

3. Answer Controllability:

- Provides parameters to influence the LLM's generation process.
- Supports setting standard generation parameters such as `temperature` (controlling randomness/creativity), `top_p` (nucleus sampling), `max_tokens` (maximum length of the generated answer), and others.

Note on Model Selection:

- Other models like `flan-t5-base`, `flan-t5-large`, `GPT2`, and `GPT2-QA` were tested during development but were found to be **not suitable** or performant enough specifically for Chinese answer generation within this system's constraints.

3.3.5 User-Friendly Chat Interface

This component provides the front-end through which users interact with the RAG system. It is designed for ease of use and a clear presentation of information.

Key Features:

1. Design and Layout:

- Features a modern User Interface (UI) design.

- Supports a responsive layout, ensuring usability across different screen sizes and devices.

2. Conversation Handling:

- Supports natural, multi-turn conversations, allowing users to engage in extended dialogues with the system.
- Includes functionality for managing the conversation history, maintaining context across turns.

3. Output Formatting:

- Supports Markdown format for generated responses.
- Ensures that formatted text (e.g., bolding, lists, code blocks) is displayed correctly for better readability.

4. Model Selection Feature:

- Allows users to choose the underlying text generation model.
- Provides a dropdown menu for easy selection.
- Currently supports switching between two modes:
 - Local DeepSeek model (referring to the 7B quantized model for local inference).
 - Ollama deployed model (referring to models accessible via an Ollama service, likely the 1.5B or similar).

5. Retrieval Results Display:

- Enhances transparency by showing the sources used for generating the answer.
- Provides a display area for reference documents or retrieved text chunks related to the answer.
- Shows the relevance score for each retrieved document or chunk, indicating its importance.
- Includes a collapsible source area feature to efficiently manage screen space while allowing users to inspect the sources when needed.

6. Conversation Management:

- Automatically saves the history of the conversation.
- Provides a user function to clear the conversation history.
- Tracks individual user sessions using session IDs to maintain separate conversation contexts.

- Storing document embedding vectors and performing fast approximate nearest neighbor search.
- Implementing semantic similarity retrieval functionality.
- Provides GPU acceleration capability.
- Supports migrating indexes to GPU via `faiss.index_cpu_to_gpu`.

3. HuggingFace Transformers

- **Usage Scenario:** Loading and inferencing pre-trained language models
- Specific Applications:
 - Loading large language models: `DeepSeek-R1-Distill-Qwen-7B`.
 - Model quantization (4-bit) to reduce memory footprint.
 - Creating text generation pipelines (`pipeline`).
 - Loading pre-trained models via `AutoTokenizer` and `AutoModelForCausalLM` interfaces.

4. Sentence-Transformers

- **Usage Scenario:** Text vectorization and semantic search
- Specific Applications:
 - Using the `all-MiniLM-L6-v2` model to convert text into 384-dimensional vectors.
 - Calculating semantic similarity between text segments.
 - Serving as the core embedding model supporting document retrieval.
 - Provides `CrossEncoder` for result re-ranking.

3.5 Performance Optimization

This section outlines the key strategies employed across the system to maximize performance, particularly focusing on achieving efficient operation on resource-constrained, consumer-grade hardware. The primary goal is acceleration throughout the RAG pipeline.

Specific Optimization Areas:

1. Model Optimization (Generation Model):

- 4-bit Quantization for the 7B Model:
 - Applies advanced quantization techniques including quantization, double quantization, and mixed precision.
 - Utilizes the NF4 format for efficient weight representation.
 - Incorporates memory optimization techniques like Memory Offloading to reduce GPU VRAM usage.
- Generation Parameter Tuning:
 - **Limits output length:** Set `max_new_tokens` to 512 to constrain response size and generation time.
 - **Controls randomness:** Configures `temperature` (set to 0.7) and `top_p` (set to 0.9) to balance creativity and coherence.
 - **Reduces repetition:** Applies a `repetition_penalty` (set to 1.1) to discourage the model from repeating phrases.

- Ollama-based Model:
 - Supports using models deployed via Ollama, which often leverage efficient packaging and hardware acceleration built into the Ollama runtime for faster inference.

2. Vectorization Optimization:

- Efficient Embedding Model:
 - Uses the `all-MiniLM-L6-v2` model with FP16 precision.
 - This model is specifically designed for speed, providing semantic embeddings approximately 4 times faster than traditional BERT-based models.

3. Retrieval Acceleration:

- Leveraging FAISS and Index Structure:
 - Employs the FAISS library for vector indexing and search.
 - Specifically utilizes the `IndexHNSWFlat` index type, which is highly optimized for fast approximate nearest neighbor search on large vector datasets.

4. Startup Optimization:

- Accelerated Model Loading:
 - Implements techniques like `Lazy Load` and `low_cpu_mem_usage`.
 - This significantly speeds up the initial loading of the large language model while minimizing the CPU memory footprint during startup.

5. Inference Optimization (Runtime):

- KV Cache Utilization:
 - Uses `use_cache` and benefits from KV (Key-Value) cache reuse during sequential token generation.
 - This prevents recalculating attention keys and values for previously generated tokens, speeding up the decoding process.

6. Prompt Engineering:

- Although primarily aimed at improving the semantic quality and relevance of the generated response, well-designed prompts can also implicitly improve perceived performance by leading to more direct and relevant answers.
- Focuses on optimizing the interaction with the generation model to yield high-quality responses.

| Acceleration Methods | Technology Name | Function |
|-----------------------------|--|--|
| Quantization Acceleration | BitsAndBytes 4bit NF4 | Significantly reduces memory usage and accelerates inference |
| Automatic Device Allocation | <code>device_map="auto"</code> | Automatically distributes the model across multiple GPUs/CPU |
| Half-Precision Inference | <code>torch_dtype=torch.float16</code> | Faster and more resource-efficient than float32 |

| Acceleration Methods | Technology Name | Function |
|--------------------------------------|-----------------------------|---|
| Offload | offload_folder="offload" | Offloads unused model parts to CPU, reducing GPU memory pressure |
| Lazy Loading + Reduced CPU Usage | low_cpu_mem_usage=True | Reduces memory usage at the start of initialization and speeds up startup |
| Cache Management | use_cache=True | Enables KV cache during generation for faster speed |
| High-frequency Sampling Optimization | top_p / temperature / top_k | Controls output quality and improves generation efficiency |

4. Performance Evaluation

We used **DomainRAG** — a Chinese QA benchmark designed for multi-document retrieval and answer generation, developed by Renmin University of China. It evaluates conversational QA, faithful QA, time-sensitive QA, and more.

4.1 Metric Evaluation

This section details the dataset and metrics used to evaluate the performance of the RAG system, covering both the retrieval and the answer generation components.

1. Evaluation Dataset

- **Dataset Used:** DomainRAG multi-doc_QA dataset.
- Characteristics:
 - A Chinese benchmark dataset specifically designed for evaluating domain-specific Retrieval-Augmented Generation models.
 - Created by the Gaoling School of Artificial Intelligence at Renmin University of China.
 - Contains seven sub-datasets covering various capabilities such as conversational QA, structured QA, faithful QA, time-sensitive QA, noisy QA, and multi-document QA.
 - Data is sourced from the admissions system website of a university in China, collected with official authorization.
 - Generated using models like ChatGPT and manually corrected during its creation process.
- Dataset Components:
 - `question`
 - `answer`
 - `positive_reference`
 - `referred_docs` (likely documents relevant to the question)

| | question | answer | positive_references | referred_docs |
|---|---|--|---|--|
| 0 | 中国人民大学政治学与行政学专业和外交学专业在满足国家战略需求方面各专注于哪些领域? | 中国人民大学政治学与行政学专业专注于国家理论、治理理论、民主理论、战争与和平、世界秩序等政治... | [学院传统优势专业。该专业聚焦国家理论、治理理论、民主理论、战争与和平、世界秩序等政治学中的... | [政治学与行政学更新时间: 2016-03-04政治学与行政学专业发展背景及主要研究领域政治学... |
| 1 | 计算机科学与技术专业与软件工程专业在培养学生方面有哪些共同点? | [都培养能够从事计算机与信息系统设计、研发、应用、管理等工作的专业人才, 毕业生都有进入国内... | [、复合、个性化的知识结构和方向, 并具备在感兴趣领域进行独立分析和深入研究的能力。学生能... | [计算机科学与技术计算机科学与技术专业培养具有扎实的数学和计算机科学与技术基础, 兼顾科学精神... |
| 2 | 赫克曼教授和加里·贝克尔教授在劳动经济学领域的贡献有哪些不同? | [赫克曼教授的贡献在于发现早期教育和发展对个人的长期成功至关重要, 并提出了著名的赫克曼方程强... | [文化乃至伦理道德等因素的影响\n\n各国具体国情的差异又使其在社会保障制度的实践中出现很... | [劳动与社会保障更新时间: 2016-03-04让人享有社会保障郑功成劳动人事学院教授、博士... |

2. Evaluation Setup

- **Model Used:** The evaluation specifically mentions using the **Ollama DeepSeek-R1-Distill-Qwen-1.5B** model for the answer generation part of the evaluation.
- **Text Length:** The context indicates that approximately 12,000 words of text were read or processed for the evaluation.

3. Metrics Used

The evaluation employs a comprehensive set of metrics divided into two main categories:

- **Retrieval Evaluation Metrics:** Assess the quality and relevance of the documents retrieved by the Retriever component.
 - **Precision:** The proportion of retrieved documents that are actually relevant to the query.
 - **Recall:** The proportion of all relevant documents in the dataset that were successfully retrieved.
 - **F1 Score:** The harmonic mean of Precision and Recall, providing a single score that balances both metrics.
 - **NDCG (Normalized Discounted Cumulative Gain):** Evaluates the ranking quality of the retrieval results, giving higher scores if more relevant documents appear higher in the ranked list.
- **Answer Evaluation Metrics:** Assess the quality, accuracy, and overlap of the generated answer compared to reference answers.
 - **Exact Match (EM):** Measures the strict percentage of generated answers that are character-for-character identical to the reference answer.
 - **F1 Score:** A token-level overlap metric that measures the similarity between the generated answer and the reference answer based on shared tokens.

- **ROUGE-1/2/L:** A set of metrics (ROUGE-1 for unigram overlap, ROUGE-2 for bigram overlap, ROUGE-L for longest common subsequence) that evaluate the overlap between the generated text and the reference text, commonly used in summarization and generation tasks.
- **BLEU:** Primarily used for evaluating the quality of machine translation, it measures the overlap of n-grams between the generated text and one or more reference translations, often used for evaluating text generation fluency and accuracy against references.

4. results

| metrics | value |
|--------------|----------------|
| Precision | 0.29 |
| Recal | 0.36 |
| F1-Retrieval | 0.31 |
| NDCG | 0.24 |
| EM | 0.0 |
| F1-Answer | 0.41 |
| ROUGE-1/2/L | 0.38、0.12、0.23 |
| BLEU | 0.043 |

5. Analysis

The system is currently in an early functional state, characterized as **"usable but with moderate performance"**, indicating it has completed the first stage of system construction and is undergoing basic validation. Key issues observed include:

- **Low relevance in the retrieval stage**
- **Good keyword coverage in generation (F1-Answer), but poor structural accuracy (EM = 0)**
- **Limited expressiveness due to the use of small-scale models**

Retrieval Stage (Retrieval Metrics)

| Metric | Value | Interpretation |
|---------------------|-------|---|
| Precision | 0.29 | Only 29% of retrieved documents are relevant, indicating a high noise level. |
| Recall | 0.36 | 36% of relevant documents are retrieved; recall is acceptable but improvable. |
| F1-Retrieval | 0.31 | A harmonic mean of precision and recall, indicating mediocre overall retrieval. |
| NDCG | 0.24 | Poor ranking quality; suggests that the reranking module (dense+sparse fusion) requires further tuning. |

Next Steps:

- Explore stronger rerankers, such as `bge-reranker-base-zh`, or interactive reranking frameworks like **colBERT**.
- Improve vector retrieval quality using more advanced models such as `bge-m3` (a high-performance multilingual embedding model).

Generation Stage (Answer Metrics)

| Metric | Value | Interpretation |
|-------------------------|--------------------|---|
| F1-Answer | 0.41 | Decent keyword hit rate in answers, but overall answer quality is average. |
| EM (Exact Match) | 0.0 | Zero exact match indicates that the model fails to precisely reproduce gold answers. |
| ROUGE-1/2/L | 0.38 / 0.12 / 0.23 | Low token-level similarity, especially ROUGE-2, indicating poor semantic flow and structure. |
| BLEU | 0.043 | Very low, suggesting minimal n-gram overlap with reference answers; poor structure or phrasing alignment. |

4.3 Speed Benchmarks

This section presents the measured performance regarding model loading and question-answer generation time, highlighting the speed characteristics of the two primary generation model options.

Evaluation Context:

- Measurements were performed using specific chunking parameters: `chunk size = 128` and `overlap = 13`.

Performance Metrics:

| Model | Init Time | Response Time |
|---|-----------|---------------|
| DeepSeek-R1-Distill-Qwen-7B 4-bit quantized model | 23s | 16s |
| Ollama DeepSeek-R1-Distill-Qwen-1.5B | 1.5s | 5.5s |

Comparison and Advantages of Ollama:

The evaluation shows that the Ollama-based model offers significantly faster performance in both initialization and overall Q&A response time compared to the locally run 7B quantized model. This is attributed to the inherent optimizations provided by the Ollama runtime:

- **Highly Optimized Loading and Inference:** Ollama provides an optimized process for loading models, resulting in quick startup times.
- **Low Latency Inference:** Inference latency is reduced through efficient caching mechanisms (such as KV cache) and potentially leveraging underlying hardware acceleration effectively.

4.4 Chunk Size Impact

Evaluation Context:

- Evaluation using the Ollama DeepSeek-R1-Distill-Qwen-1.5B
- modelExternal text corpus (approximately 20,000 words) includes legal-related entries for testing.
- Retain the top 5 results.

| chunk size | answer time (s) | response quanlity |
|---------------------|-----------------|--|
| 128 (overlap=13) | 5.5 | Best: When the question and the retrieval content are highly repetitive, key sentences can be retrieved, but the answer is not complete. For example, when asking: "What punishment will be imposed for hiding, transferring, purchasing, selling on behalf of, or otherwise concealing or hiding property known to be related to espionage?" The response does not fully follow the retrieval results.When the question is too short, it often fails to retrieve detailed content. For example, asking "When is the law enforced?" cannot retrieve the correct content, but asking "On which date will the law be enforced?" retrieves the relevant content. |
| 256 (overlap=26) | 7.53 | Secondly: The retrieval results are too long, causing the model to forget the original question. For example, when asking "What is Article 64 of the Anti-espionage Law of the People's Republic of China?", the response is "Article 41 is...". |
| 512 (overlap=52) | 10.6 | Worst: When the question and the retrieval content are highly repetitive, key sentences cannot be retrieved. |

5. Discussions

The results show that hybrid retrieval and reranking significantly enhance answer quality, even under tight hardware constraints. While EM remains low, F1 and ROUGE scores indicate reasonable semantic consistency. The chunking strategy impacts both speed and answer relevance, suggesting further tuning opportunities.

Limitations include:

- Poor performance on short or ambiguous queries.
- Incomplete answers when chunking truncates necessary context.

6. Conclusion and Future Work

SeekMindRAG showcases how hybrid RAG systems can be efficiently deployed on consumer-grade GPUs. Future improvements will include:

- Incorporating multilingual capabilities
- Enhancing factual consistency and EM scores
- Expanding to graph-based RAG via entity relation extraction (GraphRAG)

- Improving UI customization and user feedback integration

References

- [1] Wang, S., Liu, J., Song, S., Cheng, J., Fu, Y., Guo, P., Fang, K., Zhu, Y., & Dou, Z. (2024). DomainRAG: A Chinese Benchmark for Evaluating Domain-specific Retrieval-Augmented Generation. arXiv preprint arXiv:2406.05654
- [2] Xiao G, et al. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. ICML 2023.
- [3] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv preprint arXiv:2005.11401.
- [4] Glass, M., Rossiello, G., Chowdhury, M. F. M., Naik, A. R., Cai, P., & Gliozzo, A. (2022). Re2G: Retrieve, Rerank, Generate. arXiv preprint arXiv:2207.06300



香港城市大學(東莞)
City University of Hong Kong
(Dongguan)

Innovating into the Future

SeekMindRAG

Group members:

Yuhe OuYang, Jie Wang, Lijun Xu ,

Xianjun Zhang, Longkang Wang

April , 2025

City University of Hong Kong (Dongguan)

In Celebration of the Establishment in Year 2024

SeekMindRAG



DONGGUAN

- 1 Runs on **consumer-grade GPUs**, integrating DeepSeek large model, hybrid retrieval, document parsing, and inference acceleration.
- 2 Provides strong support for **Chinese semantic search** and question answering.
- 3 Combines cost-effectiveness with robust performance for complex question-answering tasks.

Table of Contents

- Background
- Core Functionality
- Performance Optimization Strategies
- Performance Evaluation
- Follow-up Plans

Background-RAG and its Application on Consumer-grade GPUs

- Traditional RAG models rely on high-performance computing resources like server-grade GPUs (e.g., A100, H100), which limits their accessibility for small businesses, research teams, and individual developers.
- With the development of lightweight large models (e.g., DeepSeek-V2, Qwen1.5-7B, Mistral) and efficient vector search tools (e.g., FAISS, Milvus), building high-performance RAG systems on consumer-grade GPUs (e.g., RTX 30/40 series) is now feasible.
- This shift makes RAG systems more affordable and accessible, offering low-barrier intelligent Q&A solutions for education, research, and small businesses, while also enabling edge deployment and local privacy computing.



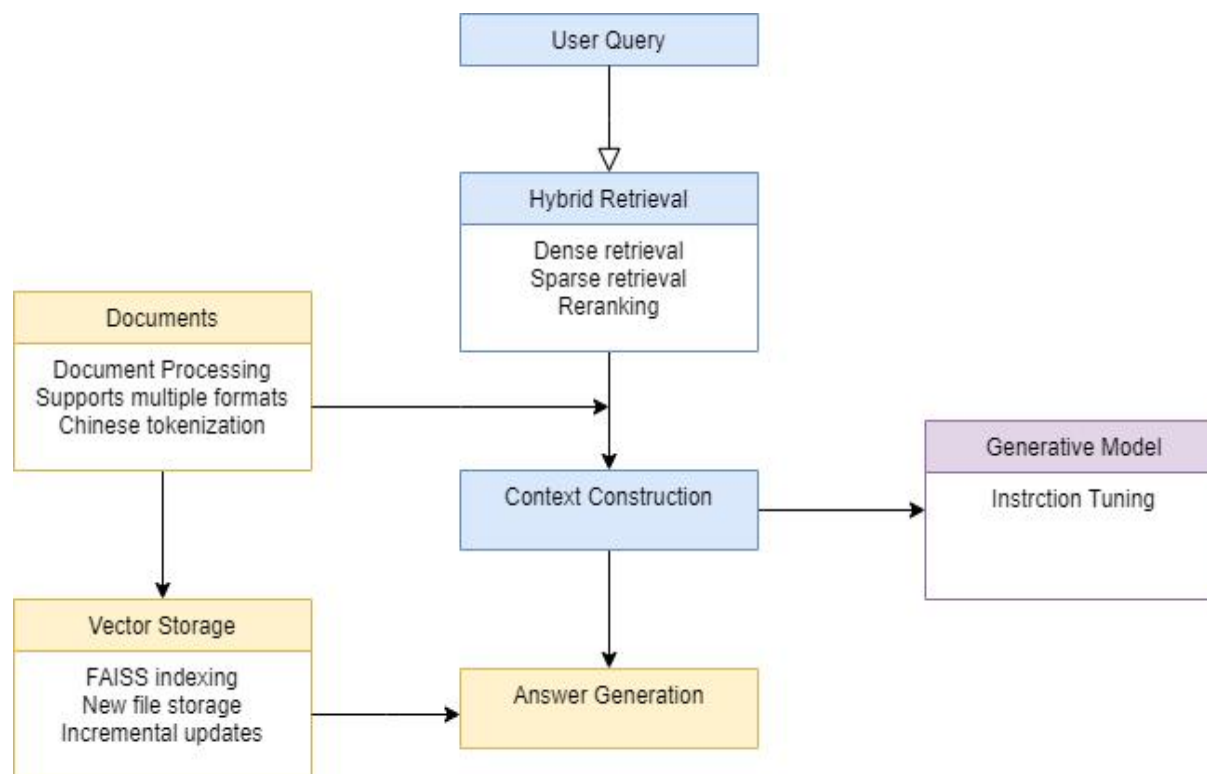
DONGGUAN

Core Functionality

System Architecture Diagram

Dependency Technologies

- **LangChain:** Large Language Model Application Framework
- **FAISS:** Vector Retrieval Engine
- **HuggingFace Transformers:** Model Loading and Inference
- **Sentence-Transformers:** Text Vectorization



System Highlights

1 Consumer-level GPU:
RTX4060, 8GB memory for operation

2 Multi-format document parsing:
Supports PDF, Word, TXT, Markdown, CSV, JSON, YAML

3 Hybrid retrieval mechanism:
Combines FAISS (dense) with BM25 (sparse) dual engines + Rerank for precise re-ranking

4 Fast local inference: Supports DeepSeek model with 4-bit quantization, requires at least 8GB memory for operation

5 Contextual Q&A: Builds context based on retrieved content to provide accurate and controllable answers

6 High configurability and scalability: Supports various indexing strategies, generation parameters, and retrieval accuracy control

7 Front-end interface support

Functional Modules — Document Processor

- **Supports Multiple Formats:** .pdf, .docx, .txt, .md, .csv, .json, .yaml
- **Chunking Strategy:** (optional chunk_size, overlap parameters)

(1)Sliding Window:

Uses sliding window technology to maintain a token buffer.

Reads text by paragraphs, tokenizes it, and adds tokens to the buffer.

When the buffer size reaches or exceeds chunk_size, it starts chunking.

(2)Smart Sentence Breaking:

Does not simply truncate by fixed length, but looks for natural sentence end positions.

Finds the sentence end position closest to chunk_size using the _find_next_sentence_end method.

Sentence end symbols include: period (.), exclamation mark (!), and question mark (?).

Continue...

- **Tokenizer:**

HanLP, a high-precision Chinese tokenizer, focused on Chinese.

- **Overlapping Handling:**

When creating the next chunk, content from the end of the previous chunk (equal to the overlap size) is retained.

This overlapping design ensures semantic coherence between adjacent chunks, improving retrieval accuracy.

- **Fault Tolerance:**

Automatically downgrades the tokenizer to ensure stability.

Uses character-level segmentation as a fallback for text where tokenization fails.

Handles possible indexing errors, ensuring safe chunking through intermediate splits.

Checks for empty buffers to avoid generating empty chunks.

Functional Modules — Vectorized Storage

- **Model:** all-MiniLM-L6-v2 (FP16 Quantization, Extremely Fast, Low Resources)
- **Index Retrieval:** FAISS (Fast Indexing for Large-Scale Vector Data), Supports Three Indexing Methods:
 - IndexFlatL2 (Euclidean Distance)
 - IndexFlatIP (Inner Product Similarity)
 - IndexHNSWFlat (Large-Scale Approximate Search, Recommended)
- **GPU Acceleration Support**
- **Dynamic Index Update:** Supports Incremental Addition of New Documents (i.e., Adding New Vectors to the Vector Database)
- **Local Secure Storage:** Storage in Local Files

Detailed Explanation of Functional Modules — Retriever

- **Hybrid Retrieval:**

FAISS Dense Semantic Retrieval:

Uses deep learning methods with semantic vector similarity.

Converts documents and queries into semantic vectors.

Calculates similarity between vectors to find relevant documents.

Uses the all-MiniLM-L6-v2 model for vectorization.

BM25 Sparse Keyword Retrieval:

A traditional term frequency-based exact match method.

Uses HanLP for Chinese word segmentation.

Computes relevance scores based on traditional keyword matching.

Best for capturing keyword matches rather than semantic similarity.

Continue...

- **Deduplication Optimization:**

Cleans and filters duplicate Chinese sentences.

- **Rerank Mechanism:**

Retrieves dense and sparse retrieval results separately.

Merges and deduplicates the two results.

Uses a cross-encoder model (cross-encoder/ms-marco-MiniLM-L6-v2) for precise re-ranking of the results.

Returns the final sorted results.

Functional Modules — Question-Answer Generation Module

- **Optional Models:**

DeepSeek-R1-Distill-Qwen-7B (4-bit local inference, high accuracy)

DeepSeek-R1-Distill-Qwen-1.5B (Ollama packaging, fast response)

- **Prompt Engineering:**

Automatically injects retrieval content, controls answer style and length

- **Answer Controllability:**

Supports setting generation parameters such as temperature, top_p, max_tokens

flan-t5-base, flan-t5-large, GPT2, GPT2-QA have been tested and are **not suitable** as Chinese generation models.

Functional Modules — Front-End UI Interface

Seekmind RAG

多文档问答系统

选择模型

DeepSeek-R1-Distill-Qwen-7B (Ollama)

清空对话历史

系统信息

检索策略:

混合检索 (向量 + BM25)

向量模型:

all-MiniLM-L6-v2

重排序模型:

ms-marco-MiniLM-L6-v2

欢迎使用基于 DeepSeek 的多文档问答系统！请输入您的问题。

用户

22:18

明知是间谍行为的涉案财物而窝藏、转移、收购、代为销售或者以其他方法掩饰、隐瞒的会受到什么处罚

助手

71.97秒

好的，我现在要回答用户的问题：“明知是间谍行为的涉案财物而窝藏、转移、收购、代为销售或者以其他方法掩饰、隐瞒的会受到什么处罚”。这个问题涉及到国家安全法中关于间谍行为的处理条款。

首先，我需要回忆一下《中华人民共和国网络安全法》的相关条款。根据提供的文档内容，第十三条规定，任何个人和组织应当组织开展反间谍安全防范宣传教育，并向全民普及反间谍知识和国家安全知识。

接下来，查看第十五条，明确指出不得非法生产、销售或者使用间谍活动所需的物品，包括存储、运输、销售、收购、代为销售涉案财物或其他需要掩盖隐瞒的行为。

因此，如果个人或组织在这种情况下进行这些行为，根据第十五条，可能会被追究刑事责任。但用户的问题中提到了“明知”，这可能会影响处罚的严重程度。全，仍然进行了这些违法活动。这可能属于更高一级的犯罪，如破坏网络安全或严重侵犯公民个人信息。

不过，根据提供的文档，没有直接提到这些行为是否构成更高级的罪行，只是提到了违反反间谍法。因此，用户的问题可能需要进一步分析，比如在何种情况下进行这些违法行为会受到怎样的处罚。

请输入您的问题...

• User-Friendly Chat Interface:

Modern UI design, supports responsive layout.

Supports multi-turn conversations and conversation history management.

Markdown format support, can correctly display formatted text.

• Model Selection Feature:

Select different generation models through a dropdown menu.

Currently supports two modes: local DeepSeek model and Ollama deployed model.

31 model.

• Retrieval Results Display:

Provides reference document display for each answer.

Shows the relevance score of the retrieved documents.

Collapsible source area to optimize interface space.

• Conversation Management:

Saves conversation history.

Provides a function to clear history.

Tracks users using session IDs.



Performance Optimization Strategies

Performance Optimization Strategies

- **Acceleration**

Model:

4-bit quantization for the 7B model:

- Quantization, double quantization, mixed precision,
- NF4 format, memory optimization (Memory Offloading)
- Limit output length to max_new_tokens=512
- Control generation randomness (temperature + top_p)
 - temperature=0.7
 - top_p=0.9
- Reduce repetition (repetition_penalty)
- repetition_penalty=1.1

Ollama-based Model

Vectorization: Uses all-MiniLM-L6-v2 + FP16, 4 times faster semantic embedding than BERT blocks.

Retrieval Acceleration: FAISS + IndexHNSWFlat

Startup Optimization: Lazy load + low_cpu_mem_usage, accelerates model loading with low memory usage

inference Optimization: use_cache + KV cache reuse

- **Prompt Engineering:** Optimizes generation model response quality

Performance Optimization Strategies

| Acceleration Methods | Technology Name | Function |
|--------------------------------------|-----------------------------|---|
| Quantization Acceleration | BitsAndBytes 4bit NF4 | Significantly reduces memory usage and accelerates inference |
| Automatic Device Allocation | device_map="auto" | Automatically distributes the model across multiple GPUs/CPUs |
| Half-Precision Inference | torch_dtype=torch.float16 | Faster and more resource-efficient than float32 |
| Offload | offload_folder="offload" | Offloads unused model parts to CPU, reducing GPU memory pressure |
| Lazy Loading + Reduced CPU Usage | low_cpu_mem_usage=True | Reduces memory usage at the start of initialization and speeds up startup |
| Cache Management | use_cache=True | Enables KV cache during generation for faster speed |
| High-frequency Sampling Optimization | top_p / temperature / top_k | Controls output quality and improves generation efficiency |

Performance Evaluation

Performance Evaluation — Metric Evaluation

- This project uses the **DomainRAG multi-doc_QA** dataset as the evaluation dataset:
 - The dataset includes **question**, **answer**, **positive_reference**, and **referred_docs**.
- DomainRAG is a Chinese benchmark dataset** specifically designed for evaluating domain-specific retrieval-augmented generation models, created by the Gaoling School of Artificial Intelligence at Renmin University of China. This dataset contains seven sub-datasets, each evaluating capabilities such as conversational QA, structured QA, faithful QA, time-sensitive QA, noisy QA, and multi-document QA. The data is sourced from the admissions system website of a university in China and was collected with official authorization. During its creation, the dataset was generated using models like ChatGPT and manually corrected.

| | question | answer | positive_references | referred_docs |
|---|---|---|---|---|
| 0 | 中国人民大学政治学与行政学专业和外交学专业在满足国家战略需求方面各专注于哪些领域？ | 中国人民大学政治学与行政学专业专注于国家理论、治理理论、民主理论、战争与和平、世界秩序等政治... | [学院传统优势专业。该专业聚焦国家理论、治理理论、民主理论、战争与和平、世界秩序等政治学中的... | [政治学与行政学更新时间：2016-03-04政治学与行政学专业发展背景及主要研究领域政治学... |
| 1 | 计算机科学与技术专业与软件工程专业在培养学生方面有哪些共同点？ | [都培养能够从事计算机与信息系统设计、研发、应用、管理等工作的专业人才，毕业生都有进入国内... | [、复合、个性化的知识结构和发展方向，并具备在感兴趣领域进行独立分析和深入研究的能力。学生能... | [计算机科学与技术计算机科学与技术专业培养具有扎实的数学和计算机科学与技术基础，兼顾科学精神... |
| 2 | 赫克曼教授和加里·贝克尔教授在劳动经济学领域的贡献有哪些不同？ | [赫克曼教授的贡献在于发现早期教育和发展对个人的长期成功至关重要，并提出了著名的赫克曼方程强... | [文化乃至伦理道德等因素的影响\n\n各国具体国情的差异又使其在社会保障制度的实践中出现很... | [劳动与社会保障更新时间：2016-03-04让人人享有社会保障郑功成劳动人事学院教授、博士... |

Performance Evaluation

- **Metrics**

- Evaluation using the Ollama DeepSeek-R1-Distill-Qwen-1.5B model
- Text length to be read is approximately 12,000 words.

- **1.Retrieval Evaluation Metrics**

Precision: The proportion of relevant documents in the retrieval results.

Recall: The proportion of relevant documents that were retrieved.

F1 Score: The harmonic mean of precision and recall.

NDCG: Normalized Discounted Cumulative Gain, considering the quality of ranking.

- **2.Answer Evaluation Metrics**

Exact Match (EM): The proportion of predicted answers that exactly match the reference answer.

F1 Score: Based on token-level overlap measure.

ROUGE-1/2/L: Evaluates the overlap between generated text and reference text.

BLEU: Evaluates the translation quality of generated text.

| metrics | value |
|--------------|------------------|
| Precision | 0. 29 |
| Recal | 0. 36 |
| F1-Retrieval | 0.31 |
| NDCG | 0.24 |
| EM | 0.0 |
| F1-Answer | 0.41 |
| ROUGE-1/2/L | 0.38、 0.12、 0.23 |
| BLEU | 0.043 |

Performance Evaluation — Response Speed

- **Evaluation using chunk size = 128, overlap = 13**

- **Average Model Initialization Time:**

DeepSeek-R1-Distill-Qwen-7B 4-bit quantized model: 23 seconds

Ollama DeepSeek-R1-Distill-Qwen-1.5B: 1.5 seconds

- **Average Q&A Response Time:**

DeepSeek-R1-Distill-Qwen-7B 4-bit quantized model: 16 seconds

Ollama DeepSeek-R1-Distill-Qwen-1.5B: 5.5 seconds

Advantages of Ollama

1. **Highly optimized model loading and inference process, allowing for quick startup.**
2. **Low latency inference:** Reduced inference latency through efficient caching mechanisms (such as KV cache) and hardware acceleration.

Performance Evaluation — chunk size

Evaluation using the Ollama DeepSeek-R1-Distill-Qwen-1.5B model

External text corpus (approximately 20,000 words) includes legal-related entries for testing.

Retain the top 5 results.

| chunk size | answer time (s) | response quantity |
|---------------------|-----------------|--|
| 128 (overlap=13) | 5.5 | Best When the question and the retrieval content are highly repetitive, key sentences can be retrieved, but the answer is not complete. For example, when asking: "What punishment will be imposed for hiding, transferring, purchasing, selling on behalf of, or otherwise concealing or hiding property known to be related to espionage?" The response does not fully follow the retrieval results. When the question is too short, it often fails to retrieve detailed content. For example, asking "When is the law enforced?" cannot retrieve the correct content, but asking "On which date will the law be enforced?" retrieves the relevant content. |
| 256 (overlap=26) | 7.53 | Secondly The retrieval results are too long, causing the model to forget the original question. For example, when asking "What is Article 64 of the Anti-espionage Law of the People's Republic of China?", the response is "Article 41 is...". |
| 512 (overlap=52) | 10.6 | Worst When the question and the retrieval content are highly repetitive, key sentences cannot be retrieved. |

Follow-up Plans

- **GraphRAG**
 - Entity Relationship Extraction, Building Knowledge Graph
- Supports More Languages

Task Division

- **all** : Topic selection, research
- **Yuhe Ouyang, Jie Wang, Lijun Xu:**
 - Multi-format text reading and chunking.
 - Retrieval module.
- **Longkang Wang, Xianjun Zhang:**
 - Generation model selection, prompt engineering, tuning,
 - Model Evaluation,
 - front-end and back-end model architecture,
 - UI interface implementation.
 - Reporting, PPT.

Thank You !

Q & A