

---

# GeoGPT-RAG Technical Report

Fei Huang<sup>1</sup>, Fan Wu<sup>1</sup>, Zeqing Zhang<sup>1</sup>, Qihao Wang<sup>1</sup>, Long Zhang<sup>1</sup>, Grant Michael Boquet<sup>1</sup> and Hongyang Chen<sup>1</sup>

<sup>1</sup>GeoGPT Team   Zhejiang Lab

GeoGPT is an open large language model system built to advance research in the geosciences. To enhance its domain-specific capabilities, we integrated Retrieval-Augmented Generation (RAG), which augments model outputs with relevant information retrieved from an external knowledge source. GeoGPT uses RAG to draw from the GeoGPT Library, a specialized corpus curated for geoscientific content, enabling it to generate accurate, context-specific answers. Users can also create personalized knowledge bases by uploading their own publication lists, allowing GeoGPT to retrieve and respond using user-provided materials. To further improve retrieval quality and domain alignment, we fine-tuned both the embedding model and a ranking model that scores retrieved passages by relevance to the query. These enhancements optimize RAG for geoscience applications and significantly improve the system's ability to deliver precise and trustworthy outputs.

GeoGPT reflects a strong commitment to open science through its emphasis on collaboration, transparency, and community-driven development. As part of this commitment, we have open-sourced two core RAG components—[GeoEmbedding](#) and [GeoReranker](#)—to support geoscientists, researchers, and professionals worldwide with powerful, accessible AI tools.

 <https://github.com/GeoGPT-Research-Project/GeoGPT-RAG>

 <https://huggingface.co/GeoGPT-Research-Project/GeoEmbedding>

 <https://huggingface.co/GeoGPT-Research-Project/GeoReranker>

 <https://huggingface.co/datasets/GeoGPT-Research-Project/GeoRAG-QA>

## 1. Introduction

Large language models (LLMs) have demonstrated impressive general-purpose capabilities, but they often fall short in specialized domains due to their tendency to generate factually incorrect or unverifiable information. Retrieval-Augmented Generation (RAG) addresses this limitation by grounding model outputs in external knowledge sources. In fields like geoscience, where precision and interpretability are essential, RAG enables models to produce context-aware and trustworthy responses. It also allows users to upload their own document collections, enabling retrieval from personalized knowledge bases tailored to specific research needs.

Beyond improving accuracy, RAG supports dynamic updates to its knowledge base, ensuring that outputs reflect the most current information available. It also promotes transparency by making the retrieval and generation process interpretable and traceable—critical for building user confidence in scientific and high-stakes applications.

RAG enhances LLM performance by retrieving semantically relevant content from external sources using similarity-based search methods. By referencing this material during generation, RAG significantly reduces hallucinations and improves factual consistency. Its integration into modern LLM pipelines has made it a foundational technique for building robust AI systems, particularly in applications like chatbots, research assistants, and domain-specific tools.

---

Initially proposed in 2020 (Lewis et al., 2020), RAG has seen rapid adoption since 2023 as its practical advantages over traditional fine-tuning have become evident. These include lower computational costs, improved interpretability, and easier adaptation to new or evolving knowledge (Gao et al., 2023).

At its core, RAG combines information retrieval with text generation in a unified workflow. It coordinates multiple components—including document retrieval, reading comprehension, response synthesis, and post-processing—into a flexible architecture. To support a range of use cases, we implemented both a lightweight (native) RAG pipeline and more advanced configurations with varying levels of complexity.

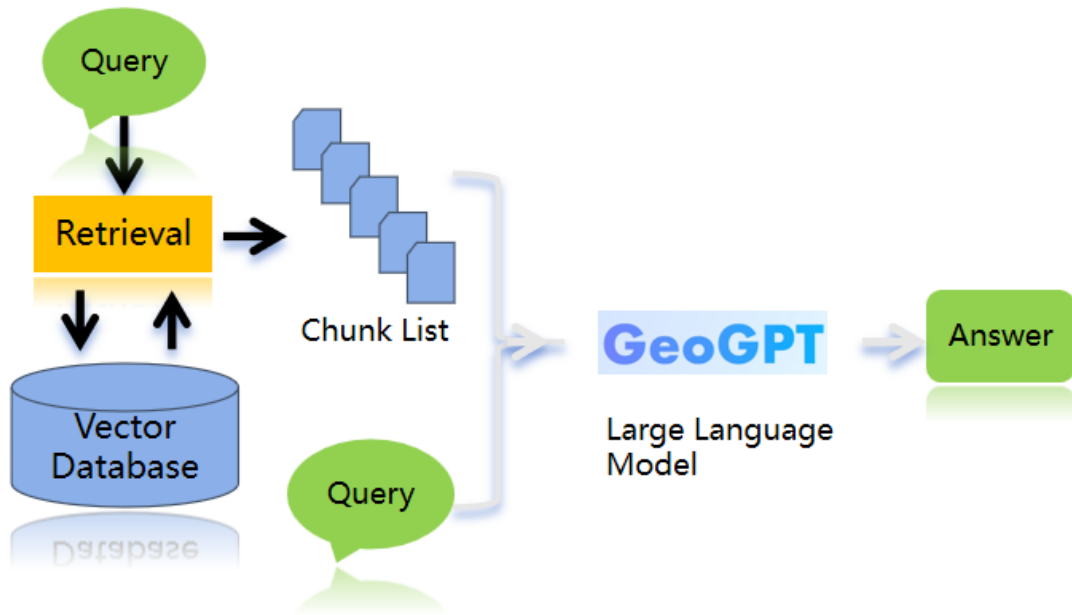


Figure 1 | Retrieval-Augmented Generation for GeoGPT

In practice, RAG retrieves relevant information and integrates it directly into the model’s prompt. This augmentation allows the model to ground its responses in verifiable context, resulting in outputs that are not only more accurate but also more transparent and easier to trust.

**Main Contributions.** Our work makes the following key contributions to the field of geographic information retrieval with large language models:

- **GeoEmbedding: the first decoder-based embedding model optimized for geoscience.** We introduce *GeoEmbedding*, the first text embedding model built on a modern decoder-only architecture (Mistral-7B) and specifically adapted to the geoscience domain. By incorporating synthetic domain-specific data during training, GeoEmbedding achieves significantly improved retrieval accuracy on geoscience queries while preserving strong general-purpose performance.
- **GeoReranker: the first reranking model tuned for geoscience relevance.** We propose and release *GeoReranker*, a domain-augmented reranking model that improves ranking precision in geoscience applications. Despite its specialization, GeoReranker remains broadly applicable, retaining high performance on general information retrieval tasks.
- **LLM-based synthetic data generation pipeline for geoscience retrieval.** To address the lack of high-quality labeled data in the geoscience domain, we design a novel synthetic data generation pipeline. This pipeline produces realistic and diverse training data by abstracting and

reformatting scholarly geoscience articles, enabling robust training of retrieval and reranking models under data-scarce conditions.

- **Data quality enhancement strategies for retrieval model training.** We implement a comprehensive data quality improvement framework, combining hard negative mining on structured data with refinement of both positive and negative samples using open-access sources. These strategies substantially improve the effectiveness and robustness of the resulting retrieval models.
- **GeoRAG-QA: a new benchmark for geoscience information retrieval.** We create and release *GeoRAG-QA*, the first large-scale benchmark designed for evaluating geoscience retrieval systems. It contains OA paper-based domain-specific documents and 938 expert-reviewed query-document pairs, providing a standardized framework for measuring performance and guiding future research in this area.

## 2. GeoGPT.RAG system UI

Integrating Retrieval-Augmented Generation (RAG) into GeoGPT significantly enhances the model's ability to support domain-specific research in geoscience. By retrieving relevant information from a curated, domain-tailored corpus—the *GeoGPT Library*—the system can generate responses that are both accurate and contextually appropriate. This is particularly advantageous for complex queries that require nuanced understanding of specialized terminology and domain-specific concepts.

The *GeoGPT Library* include corpus from open access geoscience papers, which was also used for GeoGPT training, the publishers and journals of the papers are listed here: [GeoGPT Training Data from Open Access Papers](#). To access GeoGPT, please visit our website for free [geogpt.zero2x.org](https://geogpt.zero2x.org).

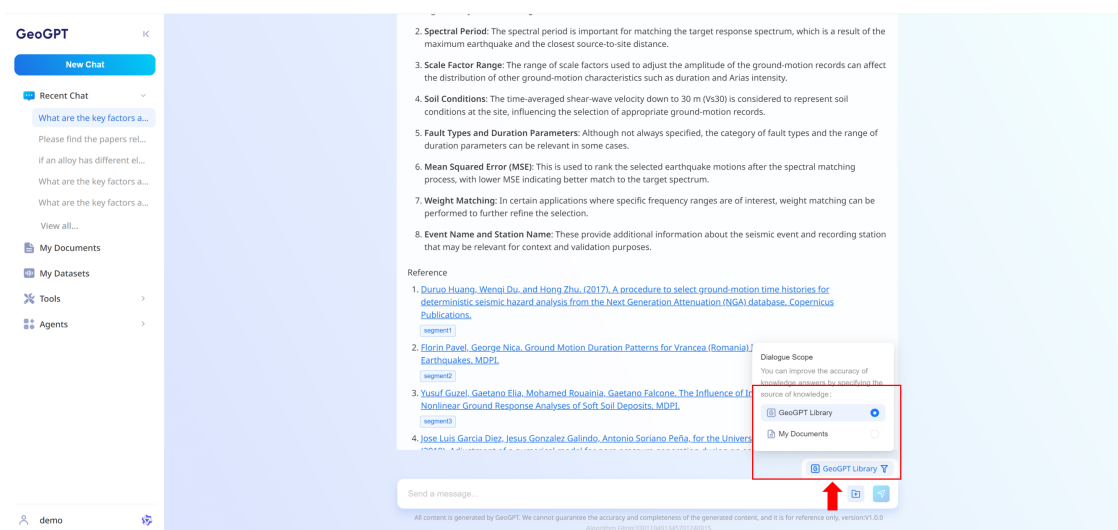


Figure 2 | ChatBox Interface for RAG with GeoGPT Library

In addition to the shared library, GeoGPT allows each user to build a personalized knowledge base through the *My Documents* feature. Users can upload their own collections of academic papers or technical documents, which are then integrated into the RAG pipeline for private, customized retrieval. This functionality effectively creates a proprietary version of GeoGPT for each user—fine-tuned to their specific areas of expertise and interests. By leveraging their own data, users can generate content that reflects their unique knowledge and perspectives, improving the relevance, depth, and accuracy of the output. The system also supports continuous updates, allowing users to refine and expand their personal knowledge base over time, ensuring that it remains current and aligned with their evolving research.

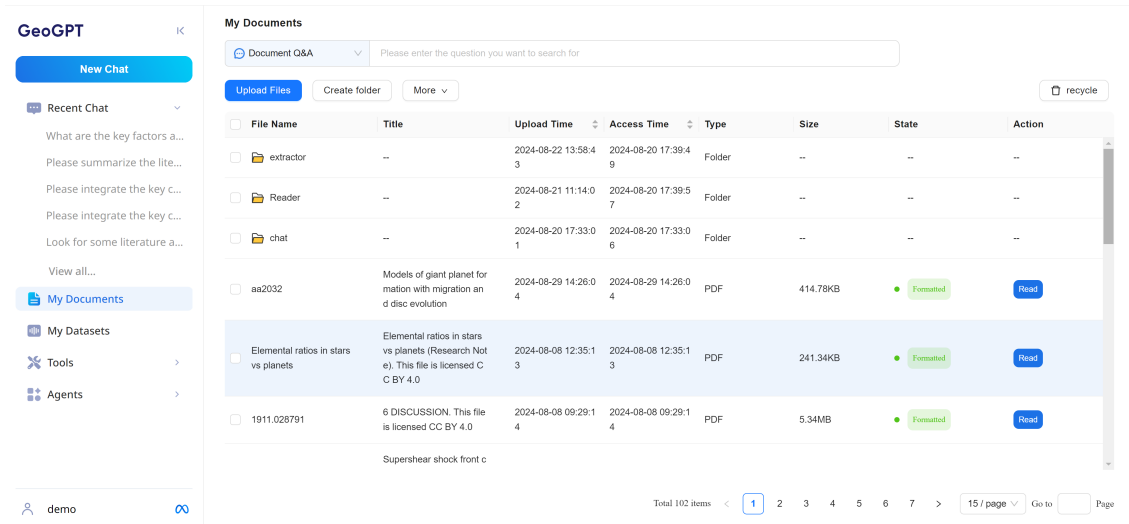


Figure 3 | "My Documents" module for documents management

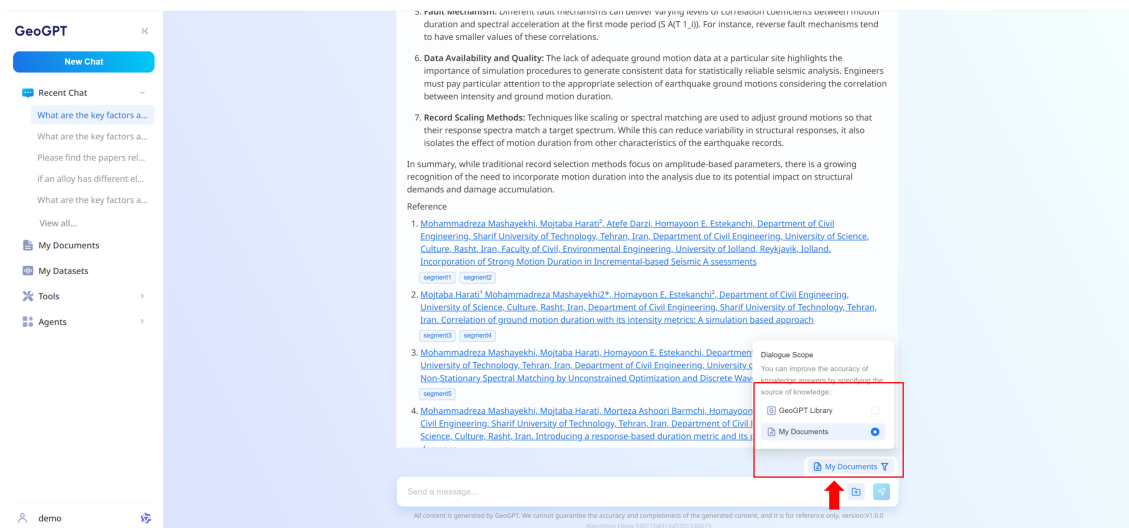


Figure 4 | RAG with My Documents

### 3. RAG Evaluation

#### a. Test Set Generation

We evaluated GeoGPT's retrieval and generation capabilities using a custom test set generated with the Test Set Generation module from RAGAS (Es et al., 2023). This test set, named GeoRAG-QA, comprises 1,000 questions derived from publicly available geoscience papers. We used gpt-4o as the underlying LLM for both question-answer (QA) pair generation and answer evaluation, and text-embedding-3-small for calculating embedding-based similarity.

To manage the computational cost of segment-wise embedding comparisons, we limited the input to a random sample of 1,000 papers from the full corpus of GeoGPT library. During question generation, some entries failed or were incomplete. After manual review, we removed 62 flawed examples, resulting in a final dataset of 938 QA items. These were categorized into four types:

- 250 single-document, fact-based QA items,

- 250 single-document inference QA items,
- 188 multi-document QA items,
- 250 conditional QA items.

Some generated questions were linked to reference answers marked as “The answer to the given question is not present in the context.” In these cases, we used GPT-4o to regenerate more appropriate reference answers, ensuring evaluation quality and consistency.

## b. Retrieval Results

Equipped with the knowledge base of GeoGPT library, our retrieval system( $top\_k = 8, score\_threshold = 0.35$ ) achieves:

Table 1 | Retrieval Results

QA Type	Question having retrieved results / All	Avg. Text Score
Single-document Simple QA	246/250	0.873
Single-document Inference QA	246/250	0.805
Multi-document QA	188/188	0.874
Conditional QA	247/250	0.820
<b>Total</b>	927/938 (0.988)	0.842

Table 2 | Top recall Results

Retrieval system	Top 1	Top 3	Top 5	Top 8	Top 32	Top 64
GeoGPT-RAG	0.908	0.945	0.95	0.959	0.966	0.969

## c. End-to-end Evaluation

### Evaluation Metric: Answer Recall

We adopt *Answer Recall* as the primary metric for end-to-end evaluation. This metric quantifies the completeness of generated answers by measuring the proportion of reference (ground-truth) statements that appear in the model-generated response (Es et al., 2023). It reflects the model’s ability to accurately retrieve and synthesize relevant information from the underlying knowledge base.

$$\text{Answer Recall} = \frac{\text{Number of ground-truth statements present in the model's answer}}{\text{Total number of ground-truth statements}}$$

Table 3 | Evaluation Results

Model	Answer Recall
GeoGPT-0630	0.529
GeoGPT-0630 + RAG	0.666

The integration of RAG significantly boosts answer completeness. Compared to the base model, GeoGPT enhanced with RAG achieves a 13.7% absolute gain in Answer Recall, highlighting the effectiveness of retrieval-based augmentation in improving factual grounding.

### Expert Evaluation on Real-World Queries

We further conducted manual evaluation on 70 domain-specific questions related to petroleum exploration. These questions were reviewed by subject matter experts to assess the factual accuracy of the generated answers. The evaluation confirms the robustness of RAG-augmented responses in real-world scenarios.

Table 4 | RAG Accuracy

Model	test data	Evaluation method	answer Accuracy
GeoGPT+RAG	70	Expert	0.857

Table 4 shows the model achieved 85.7% accuracy on expert-reviewed questions, confirming the practical effectiveness of RAG-enhanced GeoGPT in specialized geoscience domains.

### d. Retrieval Augmented Fine Tuning (RAFT) Evaluation

To further improve the model’s ability to leverage retrieved content, we incorporated task-specific RAG examples into the supervised fine-tuning (SFT) stage. This strategy—termed **Retrieval-Augmented Fine-Tuning (RAFT)**—provides the LLM with explicit examples of how to integrate retrieved content during response generation.

Table 5 | RAG QA Recall

Model	In-Domain Recall	Out-Domain Recall
GeoGPT without RAG-training data	69.72	42.04
GeoGPT with RAG-training data	78.12	46.49
<b>Advance</b>	12.05%	10.59%

The inclusion of RAG-style data during the supervised fine-tuning phase—referred to as **Retrieval-Augmented Fine-Tuning (RAFT)**—led to substantial performance gains. During RAFT, we introduced task-specific examples that demonstrated how retrieved content should be integrated into responses. These examples taught the model to align its generation more effectively with supporting evidence, reinforcing retrieval-aware reasoning during training.

We evaluated the impact of RAFT on both in-domain (geoscience) and out-of-domain (general) question-answering tasks. As shown in Table 5, the model trained with RAFT achieved an in-domain recall of 78.12%, compared to 69.72% without RAG-style training—a relative improvement of 12.05%. For out-of-domain tasks, recall increased from 42.04% to 46.49%, a 10.59% gain. These improvements suggest that RAFT enhances the model’s ability to integrate retrieved information into its outputs, leading to more complete, grounded, and contextually accurate responses.

Overall, the RAFT approach proves highly effective for reinforcing retrieval-augmented behavior during training. It not only boosts domain-specific performance but also improves generalization, making GeoGPT more reliable across both targeted and open-ended information retrieval tasks.

## 4. RAG Optimization & Model Fine-tuning

To maximize the effectiveness of Retrieval-Augmented Generation (RAG) in GeoGPT, we implemented a multi-stage optimization process. The architecture, illustrated in Figure 5, supports both batch processing of uploaded PDF files and real-time online retrieval. Our optimization strategy focuses

on three core components: text segmentation, embedding model fine-tuning, and reranker model fine-tuning.

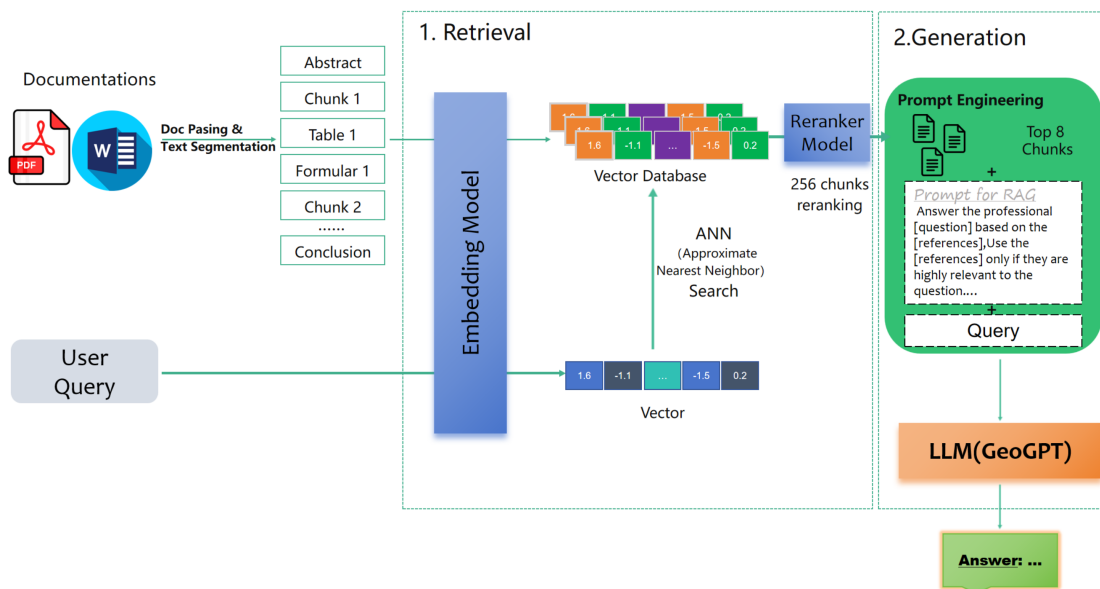


Figure 5 | RAG Processing Architecture

### a. Text Segmentation

Information chunking plays a critical role in enabling effective retrieval within RAG systems. By dividing lengthy documents into semantically coherent segments, we ensure that each chunk can be independently indexed, retrieved, and interpreted by the language model. This step is particularly important for scientific documents, which often contain dense technical language and long paragraphs with multiple concepts.

To implement chunking, we developed a semantic segmentation pipeline tailored for long-form scientific texts. For input passages exceeding 512 tokens—the typical limit for embedding models—we apply a multi-stage process:

- First, the input text is tokenized into individual sentences using NLTK's sentence segmentation toolkit. This creates a candidate list of sentence boundaries.
- Next, we apply a BERT-based Next Sentence Prediction (NSP) model to all adjacent sentence pairs. The NSP model outputs a probability score (logit) indicating the likelihood that the second sentence naturally follows the first.
- We then normalize these logits and identify the optimal split points by selecting the boundaries with the highest confidence where the predicted NSP label is 1 (i.e., coherent sentence continuation).
- Once optimal split points are selected, we segment the text accordingly. If any resulting chunk still exceeds the 512-token threshold, we recursively apply the same NSP-based segmentation strategy until all chunks are within the token limit.

This method preserves semantic continuity while adhering to token constraints, avoiding arbitrary cuts that could distort meaning. Figure 6 illustrates the final output of this segmentation strategy applied to long paragraphs, showing the alignment between sentence boundaries and token count thresholds.



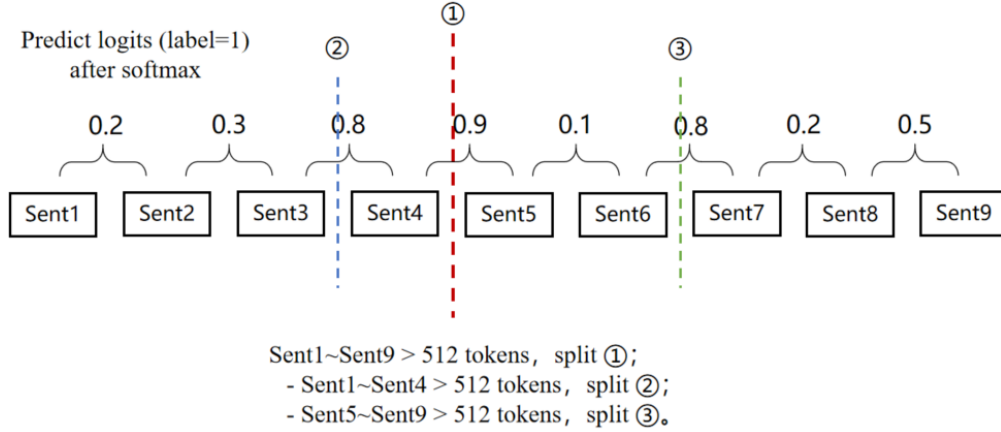


Figure 6 | Segmentation with 512 tokens

## b. Training Data Generation & Enhance RAG capability for Large Language Model

### i) SFT Data Synthesis in Geo Science

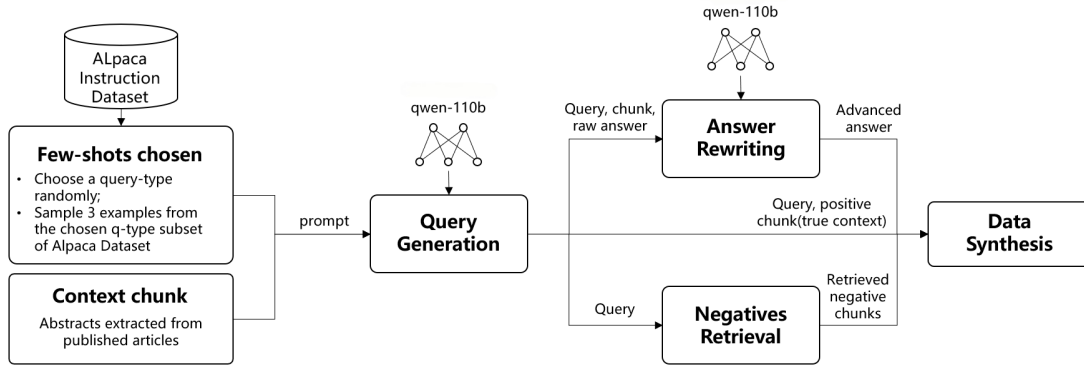


Figure 7 | Data Synthesis Process

To generate a diverse set of academic questions in Geoscience, we selected abstracts from one million publicly available papers as the context. We used the Alpaca instruction set from (Mecklenburg et al., 2024) as a few-shot sampling library, categorizing the instruction set by question types. When constructing the prompt, we first selected a question type randomly and then sampled three instruction data points from the corresponding subset to inject as few-shots into the prompt, guiding the generation of questions for that specific type. This approach increases the variety of question types in the data.

Once the initial generation was complete, we applied an Answer Rewriting Prompt to enhance the completeness and clarity of the answers. Additionally, for each question, we retrieved relevant text chunks from a knowledge base built on all publicly available papers, incorporating these as "negatives" to introduce a certain level of noise. As a result, each data context included both the original context chunk (positive) and supplementary chunks from retrieval, thereby strengthening the model's ability



to discern relevant information.

To further refine the model’s ability to distinguish textual information, we added a portion of data (approximately 7:1) where only less relevant, negatively correlated chunks were selected as the context in the retrieval process. In these cases, the answer was reformulated to "Sorry. I cannot find the answer based on the context." as mentioned in (Liu et al., 2024).

**Query-Generation Prompt:** Instruction: Given the next [document], create a [question] and [answer] pair that are grounded in the main point of the document, don’t add any additional information that is not in the document and [use prompt by different query type of Table 6]. The [question] is by an information-seeking user and the [answer] is provided by a helping AI Agent. Refer to the following question format and corresponding answers. Your output should consist solely of question-answer pairs.

[question] {few-shot 1 question}

[answer] {few-shot 1 answer}

.....

[document]: contexts

### Response:

Table 6 | Query type prompt categories

What	Which	Who/Whose	When	Where	How	Why	General question	Imperative question
The question should use [q_word] . . . to ask							Please ask in general form.	Use imperative sentences to prompt the text.

**Answer-Rewriting Prompt:** Here is a task involving RAG (Retrieval Augmented Generation) for question answering. I will provide you some documents(denoted as [References]), a question (denoted as [Query]) related to the documents, and the corresponding original answer (denoted as [Short Answer]). You are required to expand the content of the answer, with the following requirements: 1. Your generated answer should contain 6 to 8 sentences. 2. Your generated answer should have exactly the same meaning as the [Short Answer] and must perfectly address the [Query] without deviating. 3. The content of your generated answer should fully utilize the content from the [References], and you must not fabricate any facts.

[References] {documents\_here}

[Query] {question\_here}

[Short Answer] {short\_answer\_here}

Please note that do not output content other than the generated new answer. Your generated new answer is

## ii) The Dataset of DPO Processing

In this study, we used a large language model (LLM) to extract salient information from each question and its corresponding answer, identifying key concepts to guide relevance assessment. We then computed a semantic relevance score between each question and the candidate text chunks retrieved by the Retrieval-Augmented Generation (RAG) system. This scoring was based on the degree of alignment between the extracted key points and the retrieved content. To ensure high-quality

training data, we filtered out low-relevance pairs and retained only those question–chunk pairs whose relevance scores exceeded a predefined threshold of 0.4. This threshold was empirically chosen to balance precision and recall in identifying contextually appropriate retrievals for use in subsequent model training.

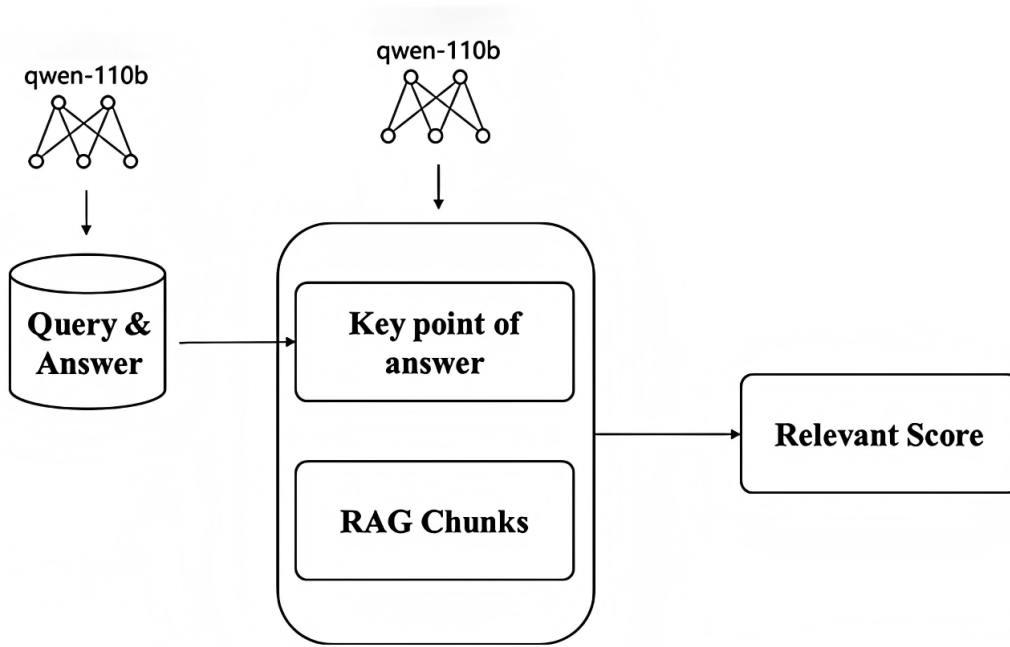


Figure 8 | Data Synthesis Process for DPO

### c. Embedding Model Fine-tuning

Embedding models are fundamental components in modern language systems, transforming text into dense, continuous vector representations. These representations support a wide range of downstream tasks, including retrieval, reranking, classification, clustering, and semantic textual similarity. Within RAG frameworks, embedding models play a central role by enabling the retrieval of semantically relevant documents from large external knowledge bases.

Historically, encoder-based architectures such as BERT and T5 have dominated the landscape of text embedding models. Recent advances in decoder-only transformer models have demonstrated competitive—and in many cases superior—performance for embedding tasks (Li et al., 2024; Meng et al., 2024). However, studies (Meng\* et al., 2024) show that converting decode-only transformer models to encode-only architectures further improves performance. In parallel, several studies (Lee et al., 2024; Wang et al., 2023) have shown that large language models can be leveraged to generate high-quality synthetic training data, enabling effective fine-tuning with fewer training steps.

In this work, we introduce **GeoEmbedding**, a domain-adapted embedding model fine-tuned from a state-of-the-art encode-only transformer converted from a decode-only transformer. To improve its in-domain retrieval capabilities, we trained GeoEmbedding using synthetic geoscientific data, specifically tailored to the linguistic and semantic patterns found in academic literature from the geography and earth sciences domains.

**Datasets.** To train our GeoEmbedding model, we used a combination of publicly available datasets spanning a range of NLP tasks, including retrieval, classification, clustering, reranking, and semantic textual similarity (STS). This diverse set of tasks enables the model to learn general-purpose em-

beddings while being robust to domain-specific requirements. We sample about 360k data from datasets including SciFact (Wadden et al., 2020), S2ORC (Lo et al., 2020), PAQ (Lewis et al., 2021), Specter (Cohan et al., 2020), and FiQA (Maia et al., 2018), etc. Additionally, we incorporate some datasets from the MTEB benchmark, such as AmazonCounterfactual, Banking77, Emotion, MassiveIntent, MassiveScenario, and ToxicConversations, etc.

Following standard practice in prior work such as SFR-Embedding-Mistral and NV-Embed, we train using only the training splits from each dataset. Where datasets include annotated negatives, we use them directly for contrastive learning. For datasets without sufficient negative samples—or none at all—we apply SimANS (Zhou et al., 2022) to mine hard negatives from the corresponding corpus.

In addition to general-purpose datasets, we generate domain-specific data for geoscience using Qwen. These synthetic examples are based on abstracts from open-access scholarly articles collected from arXiv under CC BY license, with proper attribution. We compile these into a dedicated training set, and apply SimANS to mine hard negatives from the domain corpus, ensuring the model learns to distinguish subtle differences in geoscientific content.

**Training.** Given the diversity of tasks across datasets, we prepend a task-specific instruction to each query to guide the model during training. For each query–document pair, we construct an *instructed query* by combining the task instruction  $I$  with the query  $q$  in the following format:

$$q_{\text{inst}} = \text{Instruct: } \{I\} \backslash \text{nQuery: } \{q\}$$

We apply this formatting only to the query; no instruction is added to the document side. To fine-tune the embedding model, we adopt instruction tuning with in-batch negatives using the InfoNCE loss function:

$$\min \mathbb{L} = -\log \frac{\phi(q_{\text{inst}}, d^+)}{\phi(q_{\text{inst}}, d^+) + \sum_{n_i \in N} \phi(q_{\text{inst}}, n_i)}$$

where  $q_{\text{inst}}$  is the instructed query,  $d^+$  is the corresponding positive document, and  $n_i$  denotes the  $i$ -th negative document in the batch. The similarity function  $\phi(\cdot, \cdot)$  is defined using a temperature-scaled cosine similarity:

$$\phi(q, d) = \exp\left(\frac{\cos(\mathbf{h}_q, \mathbf{h}_d)}{\tau}\right)$$

where  $\mathbf{h}_q$  and  $\mathbf{h}_d$  are the vector embeddings of the query and document, respectively, and  $\tau$  is a temperature hyperparameter controlling the sharpness of the similarity distribution.

To enable parameter-efficient training, we apply Low-Rank Adaptation (LoRA) (Hu et al., 2021), setting the rank  $r = 8$  and scaling factor  $\alpha = 16$ . The model is fine-tuned from **Mistral-7B-v0.1** (Jiang et al., 2023), a 7B decoder-only transformer pretrained model that achieved state-of-the-art performance in 2023. We fine-tune the model over 4096 gradient accumulation steps with a learning rate of  $1 \times 10^{-5}$ .

We convert the Mistral model into an encode-only architecture by removing its causal attention masks. Additionally, we apply Matryoshka representation learning during training to enhance the quality of low-dimensional embeddings.

**Evaluation.** We evaluate our embedding model on both general-purpose and domain-specific benchmarks to assess its performance across retrieval tasks.

For general evaluation, we use the **MTEB(eng)** benchmark, which includes a diverse set of retrieval datasets. Table 7 summarizes our results. We compare our model against two strong baselines—SFR-Embedding-Mistral and NV-Embed-v2—using scores replicated from the official MTEB leaderboard<sup>1</sup>

<sup>1</sup><https://huggingface.co/spaces/mteb/leaderboard>

Table 7 | Evaluation on MTEB tasks

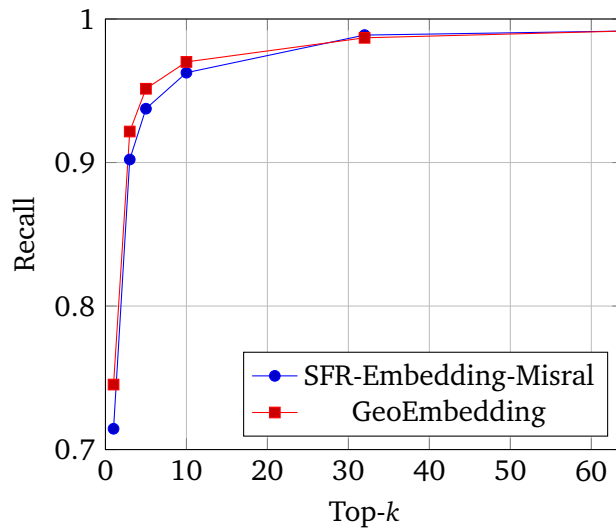
	Classification	Clustering	PairClassification	Reranking	Retrieval	STS	Summarization	Overall
<b>SFR-Embedding-Mistral</b>	80.47	54.93	88.59	<b>50.15</b>	59.33	84.77	36.32	69.31
<b>NV-Embed-V2</b>	87.19	47.66	<b>88.69</b>	49.61	<b>62.84</b>	<b>83.82</b>	<b>35.21</b>	69.81
<b>GeoEmbedding</b>	<b>89.67</b>	<b>56.50</b>	82.51	48.32	60.91	80.60	30.43	<b>70.21</b>

Model	Zero-shot	Memory Usage (MB)	Number of Parameters	Embedding Dimensions	Max Tokens	Mean (Task)	Mean (TaskType)	Classification	Clustering	Pair Classification
BELLE	NA	27125	7B	4896	32768	73.59	67.95	98.06	58.28	88.74
gemini-embedding-exp-03-07	95%	Unknown	Unknown	3872	8192	73.38	67.67	98.05	59.39	87.78
jasper_en_vision_language_v1	56%	3882	1B	8960	131072	71.41	66.65	98.27	60.52	88.14
gte-Qwen2-7B-instruct	NA	29040	7B	3584	32768	70.72	65.77	88.52	58.97	85.90
SFR-Embedding-2_B	85%	13563	7B	4896	32768	69.82	65.31	98.54	59.39	88.89
NV-Embed-v2	56%	14975	7B	4896	32768	69.81	65.00	87.19	47.66	88.69
Ling-Embed-Mistral	95%	13563	7B	4896	32768	69.80	65.29	83.00	54.07	88.44
text-embedding-005	95%	Unknown	Unknown	768	2048	69.68	64.77	86.03	51.91	87.62
text-embedding-004	95%	Unknown	Unknown	768	2048	69.53	64.82	86.03	51.52	87.65
stella_en_1.5B_v5	56%	5887	1B	8960	131072	69.43	65.32	89.38	57.06	88.02

Figure 9 | MTEB (eng) benchmark as of April 22, 2025.

(visualized in Figure 9). Despite being fine-tuned on synthetic in-domain data, our model retains competitive performance across general retrieval benchmarks, demonstrating robust generalization ability.

To evaluate in-domain performance, we constructed a dedicated geoscience retrieval benchmark using **GeoRAG-QA**. This benchmark includes synthetic query–document pairs generated using Qwen-110B and a retrieval corpus of GeoGPT library. As shown in Figure 10, our fine-tuned model achieves substantial gains in retrieval accuracy across all top- $k$  settings—from Recall@1 through Recall@10—demonstrating its effectiveness in domain-specific information retrieval.

Figure 10 | Top- $k$  recall on GeoRAG-QA benchmark

#### d. Reranker Model Fine-tuning

The reranker is a crucial component in Retrieval-Augmented Generation (RAG) systems. It refines the initial set of retrieved candidates by reordering them based on their semantic relevance to the query. Unlike embedding models, which encode queries and documents independently into fixed-length

vectors, the reranker jointly processes the query and document as input and directly predicts a relevance score. By modeling token-level interactions between the query and document, the reranker captures richer semantic relationships, resulting in improved ranking of the top candidates. Given a query-document pair, the model outputs a raw relevance score, which is typically passed through a sigmoid function to produce a normalized value in the range  $[0, 1]$ .

**Model and Training methods.** We adopt an industry-standard cross-encoding scheme for reranking and use the BGE-M3 model (Chen et al., 2024) as our base. BGE-M3 is known for its strong generalization capabilities and is supported by a robust open-source ecosystem, making it well-suited for real-time applications.

During training, we feed both positive and negative samples corresponding to each query into the model simultaneously. To optimize ranking performance, we apply the Multiple Negative Ranking (MNR) loss, defined as:

$$\mathcal{L}_{\text{MNR}} = -\frac{1}{N} \left( \sum_{i=1}^C \log(p_{\text{neg}_i}) - \log(p_{\text{pos}}) \right)$$

where  $N$  is the total number,  $C$  is the number of negative samples per query,  $p_{\text{pos}}$  is the predicted relevance score for the positive document, and  $p_{\text{neg}_i}$  is the score for the  $i$ -th negative document. This loss encourages the model to rank the positive document higher than its negative counterparts, thereby improving retrieval precision in downstream tasks.

**Dataset and Quality Improvement.** To maximize the reranker’s performance in the geoscience domain, we curated a large training corpus composed of both open-source and domain-specific ranking pairs. The domain-specific data were synthesized using Qwen, conditioned on abstracts from open-access scholarly articles in geography sourced from arXiv.

After assembling all positive query–document pairs, we applied SimANS (Zhou et al., 2022) to mine hard negative samples from the candidate corpus, thus forming a balanced and challenging training dataset. Since both synthetic and open-source data were generated in batches, individual data points may vary in quality. Even a small number of noisy or biased examples can significantly degrade reranking performance. To address this, we implemented a data quality filtering process to improve the reliability of training examples. For each positive pair, we used the LLaMA3 model to assess answer quality based on a set of predefined criteria. Each sample was assigned a quality score ranging from 0 to 3, following the standards detailed in Table 8. Only samples with sufficient quality were retained for training. Data with a quality label of 0 is excluded from training.

Table 8 | data quality standards

quality label	quality standards
3	completely correct, detailed, and covers all key information with clear and easy-to-understand expression.
2	mostly correct, covers most key information, and is clearly expressed but with minor shortcomings.
1	generally correct but has some errors or omissions, with mostly clear expression.
0	significant errors or omissions, lacks sufficient detail, and is not clearly expressed.

We ultimately utilized diverse types of paired data to train the model. Open-source data included:

QA pairs: DuReader (He et al., 2017), NQ, LCQMC (Liu et al., 2018), HotpotQA, etc.

Retrieval-oriented pairs: T2Ranking (Xie et al., 2023), mMARCO, MS MARCO, etc.

Similarity task pairs: NLI (Koreeda and Manning, 2021), NLI-zh, AFQMC (Lee et al., 2022), PAWSX (Yang et al., 2019), etc.

Multilingual data: Mr. TyDi.

In total, we incorporated over 360,000 open-source data pairs, along with 30,000 synthetically generated QA pairs specifically tailored for geological science applications.

For training with contrastive objectives, negative sample quality plays a critical role in reranker performance. Whenever available, we retain the original negative samples provided within each dataset. However, many datasets either lack negatives entirely or contain an insufficient number of challenging examples. To ensure training robustness, we construct a set of at least 16 negative samples per query for all datasets.

When additional negatives are required, we use a weak retriever to select hard negative candidates from a large-scale public Wiki corpus or domain-specific geoscience libraries curated from scholarly literature. We then apply the SimANS sampling strategy (Zhou et al., 2022) to prioritize harder examples from the candidate pool. Each negative sample  $neg_x$  is assigned a sampling probability based on the squared difference between its retriever score and that of the corresponding positive sample:

$$\mathcal{F}_{\text{Sample}}(neg_x) = \exp\left(-\sigma \cdot (neg_{\text{score}} - pos_{\text{score}})^2\right)$$

where  $neg_{\text{score}}$  and  $pos_{\text{score}}$  are the scores returned by the weak retriever, and  $\sigma$  is a task-specific hyperparameter that adjusts the sharpness of the sampling distribution.  $\mathcal{F}_{\text{Sample}}(neg_x)$  represents the probability of selecting  $neg_x$  as a training negative.

We adopt different  $\sigma$  values to reflect the difficulty characteristics of various dataset types. For question-answering datasets—where semantic precision is critical—we use a sharper distribution with  $\sigma = 1$ , focusing the model on distinguishing subtle contextual differences. For reranking tasks and semantic similarity datasets, where broader generalization is beneficial, we use a more relaxed sampling distribution with  $\sigma = 3$ .

By ensuring a consistent quantity of informative and diverse negative samples, this strategy strengthens the model’s ability to discriminate between highly similar documents and improves ranking accuracy in both general and domain-specific settings.

**Evaluation.** We adopted a dual evaluation approach assessing both domain-specific and general capabilities. For domain-specific performance, we conducted top-k recall tests using domain-specific datasets, with the results presented in Table 2. For general capabilities, we evaluated the model on the BEIR(Thakur et al., 2021) benchmark dataset, which includes 16 rank evaluation datasets from different domains. Under the condition that all retrievers used BM25, our model outperformed open-source state-of-the-art model in general retrieval tasks as well.

Table 9 | BEIR General Benchmark metric

BEIR metric	bge-reranker-v2-m3	GeoReranker
Map@3 average	0.31295	0.317110667
NDCG@10 average	0.468298125	0.477822667

## 5. RAG Engineering

### a. Vector Database

The vector database used in this RAG system is deployed via Zilliz Cloud, a fully managed service provided by Zilliz, the original creators of Milvus. Zilliz Cloud offers a deeply optimized, out-of-the-box Milvus experience, combining high performance with improved cost-effectiveness and user-friendly deployment. Specifically designed for production-level use, Zilliz Cloud enables application



deployment across multiple public cloud environments worldwide, while ensuring enterprise-grade data security and system elasticity. Compared to the open-source Milvus, Zilliz Cloud provides superior support for cloud-native features, scalability, and operational simplicity. A detailed comparison between the two is provided in Table 10.

Table 10 | Vector Database: Milvus VS Zilli Cloud

	Milvus	Zilliz Cloud
<b>Fully Managed</b>		
Milvus clusters that are fully managed and automated with zero operations required.	NO	YES
<b>High Availability</b>		
99.95% uptime SLA with built-in failover to ensure your Milvus clusters are always available.	NO	YES
<b>Elastic Scaling</b>		
Easily scale up to a billion-scale or down without the need to over-provision infrastructure.	NO	YES
<b>Different Machine Types for Best Price Performance</b>		
Provides two compute unit (CU) types: performance-optimized CU and capacity-optimized CU to fit different use cases.	NO	YES
<b>Infinite Storage</b>		
Cost-effectively store data at any scale without the need to increase compute resources.	NO	YES
<b>Cloud UI</b>		
User-friendly GUI to easily manage and monitor your Milvus clusters at any scale in the cloud.	NO	YES
<b>Resource monitors</b>		
Get automatic notifications to avoid service overloading.	NO	YES

The RAG system maintains two types of vector databases: a *public library* and a *user library*. The public library is designed for a broad community of scientific researchers and provides professional responses based on academic literature and Wikipedia-derived content. The user library, by contrast, supports personalized document-based question answering. In this setting, users upload their own documents, which the system automatically shards, vectorizes, and stores. This enables users to query their own content in a retrieval-augmented manner.

To support efficient querying in the user library, the system utilizes the *Partition* feature of Zilliz Cloud. A Partition is a logical component derived from a Collection and allows the physical storage to be divided into multiple segments, each referred to as a Partition. By scoping queries to a specific Partition rather than scanning the entire Collection, the system significantly improves query efficiency. In our implementation, each user’s data is stored in its own Partition, using the user ID as the Partition key.

Currently, the public library contains approximately 15 million vector entries, while the user library comprises around 3 million entries. Zilliz Cloud provides millisecond-level query latency and ultra-high recall rates, ensuring that the RAG system can deliver fast and accurate retrieval for both general-purpose and user-specific queries.

## b. Batch Processing of PDF Files from Papers and Books

Constructing a comprehensive and high-quality knowledge base is essential for enabling accurate and efficient retrieval in large language model systems based on RAG. PDF files—particularly academic papers, technical books, and scanned documents—are among the most valuable data sources due to their rich textual content and embedded metadata. This section outlines the technical workflow for batch processing PDF files, encompassing file parsing, data extraction, and vectorization for semantic indexing.

### i) File Parsing

**Text Extraction:** Optical Character Recognition (OCR) is applied to recognize and extract text from image-based or scanned PDFs. For digitally generated PDFs, native extraction tools such as PDFPlumber or PyMuPDF are preferred to maintain structural fidelity and avoid OCR errors.

**Layout Analysis:** To preserve document semantics, the layout of each page is analyzed to identify the



spatial organization of elements such as body text, images, captions, tables, equations, and footnotes. Tools like LayoutParser can assist in detecting layout regions and determining content flow across multi-column formats.

**Metadata Extraction:** Bibliographic metadata—such as author names, paper titles, publication dates, DOIs, and journal information—is extracted using a combination of embedded XMP metadata and heuristic parsing of document headers. These metadata fields are used to support indexing, filtering, and citation tracking within the knowledge base.

#### *ii) Data Extraction*

After parsing, the system performs structured data extraction and cleaning:

**Text Cleaning:** The raw extracted text often contains noise such as page headers, footers, watermarks, or extraneous whitespace. These artifacts are removed using rule-based filters and page-aware segmentation. Empty or near-empty pages are discarded.

**Structured Data Extraction:** Tabular data is detected and converted into machine-readable formats such as CSV. Paragraphs are segmented and mapped to their logical context (e.g., sections or chapters) based on layout cues and lexical markers. This facilitates document reconstruction and targeted retrieval.

**Key Information Identification:** Important content segments such as abstracts, introductions, conclusions, author affiliations, and keywords are automatically identified using pattern matching and model-based classifiers. These sections can later be prioritized during ranking or retrieval.

#### *iii) Vectorization*

In the final stage, the cleaned and structured content is prepared for integration into the vector database:

**Data Formatting:** Extracted content and associated metadata are serialized into a standardized format, such as JSON or structured key-value pairs, which aligns with the schema expected by the downstream indexing system.

**Text Chunking:** To maintain semantic coherence while accommodating input length limitations of embedding models, long text blocks are segmented into overlapping or context-aware chunks. Chunking strategies are optimized to preserve sentence boundaries and thematic continuity.

**Data Ingestion:** Each chunk is passed through a pretrained embedding model to generate dense vector representations. These vectors are indexed in the vector database (e.g., Zilliz Cloud or Milvus), along with their corresponding metadata. This enables fast, semantic search during inference and supports scalable document-level and passage-level retrieval.

### **c. Online Chat**

To enable interactive question answering grounded in a knowledge base, the system adopts a RAG framework. This approach enhances the performance of LLMs by incorporating relevant external information into the generation process. The online chat pipeline consists of the following steps:

#### *i) Query Vectorization*

When a user submits a query, the system first encodes it using a pre-trained embedding model. This transforms the natural language input into a high-dimensional vector representation that resides in the same semantic space as the knowledge base entries.

### *ii) Retrieval from the Vector Database*

The system searches the vector database using the query vector to identify the top- $N$  most similar text chunks. This retrieval step is based on vector similarity (e.g., cosine similarity or inner product) between the query and document embeddings.

### *iii) Scoring and Ranking*

The  $N$  retrieved chunks are passed to a ranking model that assigns relevance scores based on their semantic alignment with the user query. The top- $K$  highest-scoring chunks are selected as final retrieval results. This ranking process helps surface the most contextually relevant passages.

### *iv) Prompt Construction*

A prompt is constructed by combining the original user query with the top- $K$  retrieved chunks. The prompt follows a structured format designed to guide the language model in generating accurate and grounded responses. Domain-specific instructions may also be included.

### *v) Answer Generation*

The constructed prompt is fed into a domain-adapted large language model, which synthesizes a response based on both the user query and the retrieved context. The final output aims to be fluent, factually grounded, and semantically relevant to the user’s question.

## **6. Conclusion**

In this technical report, we introduced a RAG-based system designed to advance geoscientific research within the GeoGPT framework. The system integrates a complete data processing workflow, robust RAG engineering, a domain-specific embedding model, a domain-specific reranker model, a data synthesis approach, and strategies for enhancing data quality. We conducted comprehensive, multi-level evaluations, all of which demonstrated significant performance improvements. To foster community development and accelerate geoscientific innovation, we have open-sourced the system code, GeoRAG-QA dataset, GeoEmbedding model, and GeoReranker model.

## References

- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL <https://aclanthology.org/2024.findings-acl.137/>.
- Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.
- Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.
- Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, et al. Dureader: a chinese machine reading comprehension dataset from real-world applications. *arXiv preprint arXiv:1711.05073*, 2017.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L’elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *ArXiv*, abs/2310.06825, 2023. URL <https://api.semanticscholar.org/CorpusID:263830494>.
- Yuta Koreeda and Christopher D Manning. Contractnli: A dataset for document-level natural language inference for contracts. *arXiv preprint arXiv:2110.01799*, 2021.
- Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*, 2024.
- Joonho Lee, Hung Q Pham, and David R Reichman. Twenty years of auxiliary-field quantum monte carlo in quantum chemistry: An overview and assessment on main group chemistry and bond-breaking. *Journal of Chemical Theory and Computation*, 18(12):7024–7042, 2022.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115, 2021.

- Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. Making text embedders few-shot learners, 2024. URL <https://arxiv.org/abs/2409.15700>.
- Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. Lcqmc: A large-scale chinese question matching corpus. In *Proceedings of the 27th international conference on computational linguistics*, pages 1952–1962, 2018.
- Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa: Building gpt-4 level conversational qa models. *arXiv preprint arXiv:2401.10225*, 2024.
- Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.447. URL <https://www.aclweb.org/anthology/2020.acl-main.447>.
- Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Wwv’18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942, 2018.
- Nick Mecklenburg, Yiyu Lin, Xiaoxiao Li, Daniel Holstein, Leonardo Nunes, Sara Malvar, Bruno Silva, Ranveer Chandra, Vijay Aski, Pavan Kumar Reddy Yannam, et al. Injecting new knowledge into large language models via supervised fine-tuning. *arXiv preprint arXiv:2404.00213*, 2024.
- Rui Meng\*, Ye Liu\*, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfr-embedding-2: Advanced text embedding with multi-stage training, 2024. URL [https://huggingface.co/Salesforce/SFR-Embedding-2\\_R](https://huggingface.co/Salesforce/SFR-Embedding-2_R).
- Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog, 2024. URL <https://blog.salesforceairesearch.com/sfr-embedded-mistral/>.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.
- David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.609. URL <https://aclanthology.org/2020.emnlp-main.609>.
- Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*, 2023.
- Xiaohui Xie, Qian Dong, Bingning Wang, Feiyang Lv, Ting Yao, Weinan Gan, Zhijing Wu, Xiangsheng Li, Haitao Li, Yiqun Liu, et al. T2ranking: A large-scale chinese benchmark for passage ranking. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2681–2690, 2023.
- Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. Paws-x: A cross-lingual adversarial dataset for paraphrase identification. *arXiv preprint arXiv:1908.11828*, 2019.
- Kun Zhou, Yeyun Gong, Xiao Liu, Wayne Xin Zhao, Yelong Shen, Anlei Dong, Jingwen Lu, Rangan Majumder, Ji-Rong Wen, Nan Duan, et al. Simans: Simple ambiguous negatives sampling for dense text retrieval. *arXiv preprint arXiv:2210.11773*, 2022.