# Improving the Bible QA RAG System with Advanced Techniques

## Multi-Agent Chain-of-Thought Retrieval

One promising strategy is to introduce a **multi-agent orchestrator** that can plan and reason through complex questions. Instead of a single-shot retrieve-and-answer, **specialized agents** collaborate on sub-tasks: e.g. a Planner (to break down the query), a Retriever (to fetch relevant verses), an Extractor (to filter key evidence), and a QA Agent (to synthesize the final answer)[[1]](https://arxiv.org/html/2505.20096v2#:~:text=collaborative%20set%20of%20specialized%20AI,methods%20across%20all%20model%20scales). These agents communicate via chain-of-thought reasoning, refining the search and answer step by step. Recent research shows this **multi-step approach dramatically improves results** on complex, multi-hop questions: even a small 8B model with such a system outperforms much larger LLMs using vanilla RAG[[2]](https://arxiv.org/html/2505.20096v2#:~:text=Notably%2C%20even%20a%20small%20LLaMA3,hop). In practice, your offline setup could run a lightweight “planner” model (for high-level thinking) to orchestrate calls to the Bible vector DB and then hand off to a stronger “answer” model for final generation[[3]](https://arxiv.org/html/2505.20096v2#:~:text=Extensive%20experiments%20on%20multi,RAG)[[4]](https://arxiv.org/html/2505.20096v2#:~:text=This%20multi,Moreover%2C%20our%20analysis%20shows). This **agentic planning** ensures ambiguous or multi-part queries are handled via **stepwise reasoning** rather than a single dense retrieval pass.

## Knowledge Graph-Augmented Retrieval (GraphRAG)

Enhancing the vector database with a **knowledge graph of biblical entities and concepts** can yield more relevant context. A knowledge graph (e.g. linking people, places, themes across verses) allows retrieval of not just textually similar verses, but **related context via relationships**[[5]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=). For example, a question about “covenants” might fetch verses about Abraham, Moses, and Jesus if those are connected through a “covenant” concept node. Graph-based retrieval excels at **multi-hop questions** spanning different books, where pure embeddings might miss connections[[6]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Why%20graphs%20matter%3A%20Vector%2Fsemantic%20search,can%20answer%20with%20greater%20accuracy). In practice, you can blend **graph traversals with vector search**: first use embeddings to find an initial verse, then pull connected verses (by shared entities or links) from the graph for a richer context[[5]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=). This **GraphRAG** approach provides a more **precise and explainable** context assembly than vectors alone, and it aligns well with the existing verse–concept links in your bible\_db. (Notably, open-source frameworks like *RAGFlow* and *LlamaIndex* already support building knowledge graphs for retrieval[[7]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=%2A%20Visual%20web%20interface%20,both%20Elasticsearch%20and%20Infinity%20for)[[8]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=Key%20features%20of%20LlamaIndex%20include%3A).)

## Hybrid Search (Vectors + Keywords)

To improve recall, consider **hybrid retrieval** that combines semantic embeddings with traditional keyword search. Dense vectors capture meaning, but they might overlook exact phrasing or rare terms (e.g. names, archaic words). A **BM25 or lexical search** on the Bible text can complement embeddings by ensuring exact matches for key terms are not missed[[9]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Lexical). Merging the results (e.g. via Reciprocal Rank Fusion) often yields better coverage[[10]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Semantic%20retrieval%20encodes%20queries%20and,meaning%20and%20the%20right%20tokens). For instance, a query “Where does the Bible mention **melchizedek**?” would benefit from lexical search to catch the rare name “Melchizedek” even if the embedding isn’t a close match. By **pairing pgVector with a keyword index**, the system can retrieve passages that have **both the right meaning and the right keywords**[[9]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Lexical), boosting overall accuracy. This hybrid strategy is open-source friendly: you can use PostgreSQL full-text search or Elasticsearch alongside pgvector.

## Enhanced Chunking and Re-Ranking of Context

Improving how verses are **chunked and ranked** before sending to the LLM can significantly boost answer quality. Ensure your Bible text is segmented in a meaningful way – for example, by per-verse or small passage – and consider **semantic chunking** (splitting on narrative boundaries or topics) rather than fixed-size chunks[[11]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=). After initial retrieval, apply a **re-ranking model** (like a cross-encoder) to the candidate verses so that the most relevant passages are prioritized in the LLM’s context[[12]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=). This second-pass re-ranker (which can be a smaller **open-source model fine-tuned for QA** relevance) often improves the quality of the top-5 results fed into the answer generator. Additionally, you can use **context distillation**: if too many verses are relevant, have a model summarize or compress them before injection[[13]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Distillation). This ensures the LLM gets a dense, relevant context within its token limit. All these steps can be implemented with open tools (e.g. Hugging Face cross-encoders for rerank, or LlamaIndex’s summarizers[[13]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Distillation)) to keep the pipeline offline.

## Query Expansion and Reformulation

Biblical queries might use language that doesn’t exactly match the text (consider differences in translation or phrasing). A **query understanding module** can rewrite or expand the user’s question to improve retrieval[[14]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Improve%20Query%20Understanding)[[15]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Retrieval%20Variant). For example, if a user asks about “charity,” the system could expand it with synonyms like *love* or *almsgiving* to catch relevant verses. Techniques like **Hypothetical Document Embeddings (HyDE)** generate a fake answer or related questions from the query and embed those[[16]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Retrieval%20Variant), bridging the gap between the question and the biblical wording. Similarly, adding **synonyms and related terms** (via a thesaurus or an LLM prompt) will increase the chance of hitting relevant verses[[15]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Retrieval%20Variant). This step can be done with small open-source models or rules (for example, a GPT-2 based rewriter or a simple WordNet synonym expander), all running locally. By **feeding the retrieval engine a richer query**, you reduce misses due to vocabulary mismatch.

## Verification and Iterative Refinement (CRAG/CoT Loops)

Even with good retrieval, it’s wise to add a **feedback loop to verify and refine** the answer. One approach is **Corrective RAG (CRAG)**: after retrieving, have the system **check if the context likely contains the answer** before final generation[[17]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=When%20strong%20retrieval%20pipelines%20return,answers%20tied%20to%20stronger%20evidence). If the retrieved verses seem off-target or insufficient, the system can trigger a second retrieval with adjusted parameters or a refined query[[17]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=When%20strong%20retrieval%20pipelines%20return,answers%20tied%20to%20stronger%20evidence). This ensures the model isn’t forced to “fill in gaps” (a source of hallucination) – instead, it fetches better support or otherwise says “I don’t know.” Another approach from research is the **Chain-of-Verification (CoV-RAG)**, which has the model explicitly **judge its own answer against the sources and revise if inconsistent**[[18]](https://arxiv.org/abs/2410.05801#:~:text=RAG,baselines%20using%20different%20LLM%20backbones). In practice, you might generate an initial answer with the Bible citations provided, then ask the model (or a second validator model) to cross-check each claim against the verses and flag errors. This leads to an iterative *retrieve → answer → verify → revise* cycle. Such self-critical chains can be run with open-source models (for example, using a smaller LLM to act as a "critic" that ensures the bigger model’s answer is fully supported). By **integrating a verify-and-refine step**, the system maintains high factual accuracy and faithfulness to the text[[18]](https://arxiv.org/abs/2410.05801#:~:text=RAG,baselines%20using%20different%20LLM%20backbones).

## Open-Source Models and Implementation Tools

Crucially, all the above methods can be implemented with open-source components. For the LLMs, you already use a specialized 12B Bible QA model – you might also experiment with powerful base models like **Llama 2** or **Qwen-7B/14B**, which are available under permissive licenses. These can serve either as the main answer generator or the reasoning orchestrator (with smaller models for the latter if needed). **Vector embeddings** can likewise come from open models (your use of Qwen’s 0.6B embedder is a good example). On the orchestration side, there are frameworks to help build these advanced pipelines: **LangChain** or **LangGraph** can manage the agent loop (planning, tool calls, stepwise reasoning)[[19]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=LangGraph%2C%20or%20LlamaIndex%20can%20help,template%C2%A0and%20host%20it%20using%20LangServe), and **LlamaIndex** offers data structures for hybrid search and graph integration[[20]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=%2A%20Flexible%20data%20connectors%20,and%20other%20data%20types%20in)[[5]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=). Tools like **RAGFlow** even combine many of these ideas (document parsing, GraphRAG, agent reasoning) in one open platform[[21]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=RAGFlow%20offers%20powerful%20features%20designed,based%20retrieval). All these are open source and can run fully offline with the right hardware. By leveraging these libraries and models, you can **upgrade the RAG system’s intelligence** without proprietary services – the system will plan its searches, retrieve more relevant scripture passages, and deliver answers with improved accuracy and supporting citations.

## References and Further Reading

* Gao et al., *“MA-RAG: Multi-Agent Retrieval-Augmented Generation via Collaborative Chain-of-Thought Reasoning,”* 2025[[1]](https://arxiv.org/html/2505.20096v2#:~:text=collaborative%20set%20of%20specialized%20AI,methods%20across%20all%20model%20scales)[[2]](https://arxiv.org/html/2505.20096v2#:~:text=Notably%2C%20even%20a%20small%20LLaMA3,hop) – Introduces a multi-agent RAG framework that outperforms standard single-step RAG, highlighting the benefit of planner and extractor agents for complex queries.
* Neo4j Developer Blog, *“Advanced RAG Techniques for High-Performance LLM Applications,”* 2025[[5]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=)[[22]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Some%20questions%20require%20stitching%20facts,hop%20returns%20weak%20evidence%2C%20expand) – Discusses GraphRAG (knowledge-graph-enhanced retrieval), hybrid semantic+lexical search, agentic multi-hop planning, and other techniques to boost RAG pipelines.
* He et al., *“Retrieving, Rethinking and Revising: The Chain-of-Verification Can Improve RAG,”* EMNLP 2024[[18]](https://arxiv.org/abs/2410.05801#:~:text=RAG,baselines%20using%20different%20LLM%20backbones) – Proposes a chain-of-thought verification step in RAG to iteratively correct retrieval errors and generation mistakes, leading to more consistent, accurate answers.
* **Open-Source Tools:** *LlamaIndex* documentation[[8]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=Key%20features%20of%20LlamaIndex%20include%3A) and *RAGFlow* features[[7]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=%2A%20Visual%20web%20interface%20,both%20Elasticsearch%20and%20Infinity%20for) for implementing graph-based retrieval, reranking, and agent orchestration in practice; *PageIndex* (Vectify AI) for a novel reasoning-based, hierarchy-oriented retrieval alternative[[23]](https://www.reddit.com/r/Rag/comments/1n1iqy3/humanlike_rag_without_vectors/#:~:text=PageIndex%20takes%20a%20different%20approach,context%20rather%20than%20matching%20embeddings). Each of these can be explored for potential integration, keeping the entire stack free and open-source.

[[1]](https://arxiv.org/html/2505.20096v2#:~:text=collaborative%20set%20of%20specialized%20AI,methods%20across%20all%20model%20scales) [[2]](https://arxiv.org/html/2505.20096v2#:~:text=Notably%2C%20even%20a%20small%20LLaMA3,hop) [[3]](https://arxiv.org/html/2505.20096v2#:~:text=Extensive%20experiments%20on%20multi,RAG) [[4]](https://arxiv.org/html/2505.20096v2#:~:text=This%20multi,Moreover%2C%20our%20analysis%20shows) MA-RAG: Multi-Agent Retrieval-Augmented Generation via Collaborative Chain-of-Thought Reasoning

<https://arxiv.org/html/2505.20096v2>

[[5]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=) [[6]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Why%20graphs%20matter%3A%20Vector%2Fsemantic%20search,can%20answer%20with%20greater%20accuracy) [[9]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Lexical) [[10]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Semantic%20retrieval%20encodes%20queries%20and,meaning%20and%20the%20right%20tokens) [[11]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=) [[12]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=) [[13]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Distillation) [[14]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Improve%20Query%20Understanding) [[15]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Retrieval%20Variant) [[16]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=,Retrieval%20Variant) [[17]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=When%20strong%20retrieval%20pipelines%20return,answers%20tied%20to%20stronger%20evidence) [[19]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=LangGraph%2C%20or%20LlamaIndex%20can%20help,template%C2%A0and%20host%20it%20using%20LangServe) [[22]](https://neo4j.com/blog/genai/advanced-rag-techniques/#:~:text=Some%20questions%20require%20stitching%20facts,hop%20returns%20weak%20evidence%2C%20expand) Advanced RAG Techniques for High-Performance LLM Applications - Graph Database & Analytics

<https://neo4j.com/blog/genai/advanced-rag-techniques/>

[[7]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=%2A%20Visual%20web%20interface%20,both%20Elasticsearch%20and%20Infinity%20for) [[8]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=Key%20features%20of%20LlamaIndex%20include%3A) [[20]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=%2A%20Flexible%20data%20connectors%20,and%20other%20data%20types%20in) [[21]](https://www.firecrawl.dev/blog/best-open-source-rag-frameworks#:~:text=RAGFlow%20offers%20powerful%20features%20designed,based%20retrieval) 15 Best Open-Source RAG Frameworks in 2025

<https://www.firecrawl.dev/blog/best-open-source-rag-frameworks>

[[18]](https://arxiv.org/abs/2410.05801#:~:text=RAG,baselines%20using%20different%20LLM%20backbones) [2410.05801] Retrieving, Rethinking and Revising: The Chain-of-Verification Can Improve Retrieval Augmented Generation

<https://arxiv.org/abs/2410.05801>

[[23]](https://www.reddit.com/r/Rag/comments/1n1iqy3/humanlike_rag_without_vectors/#:~:text=PageIndex%20takes%20a%20different%20approach,context%20rather%20than%20matching%20embeddings) Human-like RAG – without vectors : r/Rag

<https://www.reddit.com/r/Rag/comments/1n1iqy3/humanlike_rag_without_vectors/>