Learning to Retrieve, Generate, and Compress: A Unified View of Efficient RAG

Faruq Brontes , Jeanie Genesis * , Zachariah Noa , Sigiwardaz Nymphodoros

Posted Date: 18 August 2025

doi: 10.20944/preprints202508.1211.v1

Keywords: retrieval-augmented generation; large language models; efficient inference; neural retrieval; End-to-End optimization; knowledge-intensive NLP; memory-augmented models; context compression; multihop reasoning; latency reduction; hybrid architectures; document indexing; grounded generation; query-adaptive retrieval; scalable AI systems

 doi:10.20944/preprints202508.1211.v1

Article

Learning to Retrieve, Generate, and Compress: A Unified View of Efficient RAG

Faruq Brontes [1], Jeanie Genesis [1,*], Zachariah Noa [2,3] and Sigiwardaz Nymphodoros [3]

[1] Department of Computer Science, Lancaster University
[2] Department of Artificial Intelligence, University of Cambridge
[3] Department of Computer Science, University of Oxford
* Correspondence: j.genesis@lancaster.ac.uk

Abstract

Retrieval-Augmented Generation (RAG) has become a foundational technique in natural language processing and AI systems, enabling large language models (LLMs) to dynamically condition on external knowledge during inference by retrieving relevant documents from large corpora. This hybrid approach enhances the factuality, transparency, and adaptability of generation by combining the parametric knowledge of pre-trained transformers with non-parametric retrieval mechanisms. Despite its growing adoption across a range of domains—from open-domain question answering and knowledge-intensive tasks to scientific research, legal reasoning, and enterprise applications—RAG presents unique computational and modeling challenges. These include latency from multi-stage pipelines, retrieval-generation misalignment, context redundancy, faithfulness issues, and scalability constraints in handling billion-scale document indexes. This survey provides a comprehensive and deeply technical overview of the emerging landscape of efficient Retrieval-Augmented Generation for foundation models. We begin with a formal characterization of the RAG problem space, presenting unified mathematical formulations of its retrieval and generation components, including probabilistic models and variational interpretations. We categorize the primary RAG paradigms, retrieval-then-generation, retrieval-as-context, retrieval-as-planning, and iterative RAG—and analyze their respective strengths and computational bottlenecks. We then delve into a detailed taxonomy of efficiency-oriented methods that improve retrieval quality, reduce inference latency, minimize memory consumption, and enable end-to-end trainability. Techniques surveyed include dense and sparse vector indexing, multi-vector compression, approximate nearest neighbor search, memory pruning, retrieval reranking, late interaction models, early exit decoding, passage filtering, and hybrid sparse-dense fusion mechanisms. To provide empirical grounding, we compile and analyze benchmark results across standard datasets (e.g., Natural Questions, TriviaQA, ELI5, FEVER, HotpotQA), architectures (e.g., DPR, FiD, REALM, RAG-Sequence, Atlas, GTR), and hardware setups. A comparative evaluation highlights the trade-offs between retrieval cost, generation accuracy, document redundancy, and interpretability across open-domain, multi-hop, and domain-adaptive settings. We also discuss auxiliary tools such as rerankers, memory controllers, and faithful decoding strategies that significantly impact performance. Beyond the current state-of-the-art, we identify key open challenges and propose a set of future research directions, including unified end-to-end optimization of retriever and generator, retrieval-aware generation objectives, dynamic and query-adaptive context selection, multimodal and multilingual retrieval integration, scalable lifelong learning architectures, and robust hallucination mitigation strategies. We emphasize the importance of developing more transparent, personalized, and controllable RAG systems that align with human expectations and safety norms. Ultimately, this survey aims to serve as both a foundational resource and a strategic roadmap for researchers and practitioners working on efficient and grounded language generation. As RAG continues to mature, it holds the promise of unlocking new capabilities for language models by enabling them to reason over explicit knowledge at scale—bridging the gap between memorization and inference, and laying the groundwork for next-generation interactive, agentic, and trustworthy AI systems.

 doi:10.20944/preprints202508.1211.v1

Keywords: retrieval-augmented generation; large language models; efficient inference; neural retrieval; End-to-End optimization; knowledge-intensive NLP; memory-augmented models; context compression; multihop reasoning; latency reduction; hybrid architectures; document indexing; grounded generation; query-adaptive retrieval; scalable AI systems

1. Introduction

The exponential growth of data across digital platforms has outpaced the ability of even the most sophisticated machine learning models to retain, process, and reason over all available information in a purely parametric fashion. In this context, *Retrieval-Augmented Generation* (RAG) has emerged as a powerful paradigm that enhances the performance of large-scale *Foundation Models* (FMs), such as large language models (LLMs), by equipping them with external retrieval mechanisms. By combining the generalization capabilities of neural generation with the precision and factual grounding afforded by retrieval systems, RAG offers a scalable and modular approach to tackling knowledge-intensive tasks including question answering, summarization, dialogue systems, and more. Foundation Models, characterized by their massive scale and pretraining on diverse corpora, have demonstrated remarkable capabilities in few-shot learning, language understanding, and zero-shot generalization. However, these models are constrained by the fixed knowledge they encode at training time, which poses limitations in handling temporal information, domain-specific content, or rare knowledge. Moreover, the cost of retraining such models to update their internal knowledge is prohibitively high. Retrieval-Augmented Generation mitigates these limitations by integrating a non-parametric memory—typically a document store or knowledge base—from which relevant context can be dynamically retrieved at inference time. This fusion enables FMs to access up-to-date and fine-grained information without requiring architectural changes or retraining, thereby decoupling knowledge acquisition from model learning [1]. RAG architectures typically follow a retrieve-then-generate framework: given an input query, a retriever fetches a set of relevant documents or passages from a large-scale corpus, which are then used by a generator (often a language model) to produce a coherent and contextually enriched response. This decoupling fosters interpretability, modularity, and adaptability. Depending on the retrieval mechanism employed—ranging from sparse methods like BM25 and TF-IDF to dense vector retrieval using dual-encoder architectures—the system can be fine-tuned to optimize either recall or semantic relevance. On the generation side, autoregressive transformers such as GPT, T5, or BART are commonly employed, capable of conditional generation over extended retrieved contexts [2]. Despite its promise, designing *efficient* RAG systems introduces several challenges. First, the retrieval step can become a bottleneck due to the high latency and computational overhead associated with searching over massive corpora, especially in low-latency applications [3]. Second, there is a tension between retrieval accuracy and computational cost: more accurate retrievers often require dense encodings and complex similarity search algorithms, while efficient retrieval may sacrifice relevance. Third, managing the interaction between retrieved context and generation remains nontrivial, particularly when dealing with noisy, redundant, or conflicting documents [4]. Other concerns include memory usage, retrieval-time scalability, real-time adaptation, and deployment constraints. Recent advancements have attempted to tackle these challenges from multiple fronts [5]. For instance, innovations in approximate nearest neighbor (ANN) search, retrieval distillation, retrieval caching, and hybrid sparse-dense methods aim to accelerate retrieval without sacrificing quality [6]. On the generation side, architectural innovations such as context-window extension, retrieval-aware decoding, and retrieval-enhanced attention have been proposed to better utilize external context. Moreover, end-to-end training techniques, joint retriever-generator optimization, and instruction-tuning have shown promise in aligning retrieval and generation behaviors for more effective RAG pipelines. This survey presents a comprehensive and structured overview of the rapidly evolving field of efficient Retrieval-Augmented Generation for Foundation Models. We aim to synthesize research across information retrieval, natural language

 doi:10.20944/preprints202508.1211.v1

processing, machine learning systems, and foundation model engineering. Specifically, we categorize existing approaches based on their architectural design, efficiency optimizations, training paradigms, and evaluation methodologies. We highlight the trade-offs and open challenges in building RAG systems that are both performant and resource-efficient [7]. Through this survey, we also delineate the key trends, foundational techniques, and promising directions that are shaping the future of RAG-enabled intelligent systems. In summary, Retrieval-Augmented Generation stands at the confluence of retrieval systems and generative modeling, offering a pragmatic and scalable alternative to purely parametric knowledge representations. As Foundation Models continue to permeate real-world applications, enhancing their capabilities with efficient, reliable, and context-aware retrieval mechanisms will be central to overcoming their limitations and achieving truly general-purpose intelligence [8].

2. Problem Formulation and Theoretical Foundations

Retrieval-Augmented Generation (RAG) systems integrate two fundamental components—retrieval and generation—into a unified framework to enhance the performance of large-scale foundation models. Formally, given a user input query $q \in \mathcal{Q}$, the goal of a RAG system is to produce a textual output $y \in \mathcal{Y}$ that is both syntactically coherent and factually grounded in an external corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$, where $d_i \in \mathcal{D}$ represents an individual document, passage, or knowledge snippet. In this section, we rigorously define the components of RAG, discuss their mathematical formulations, and explore the theoretical underpinnings governing their design and optimization.

2.1. General Framework

At a high level, RAG models implement the following decomposition of the conditional probability of the output y given the input query q:

$$P(y \mid q) = \sum_{d \in \mathcal{D}} P(y \mid q, d) P(d \mid q) \tag{1}$$

This formulation assumes a two-stage process:

1. **Retrieval Stage:** Compute the retrieval distribution $P(d \mid q)$, which models the probability that document $d \in \mathcal{D}$ is relevant to the query q.
2. **Generation Stage:** Condition the generative model on both the query q and retrieved documents d, and estimate the conditional distribution $P(y \mid q, d)$.

This decomposition enables the integration of external knowledge into the generation process without modifying the parametric model's internal weights.

2.2. Retrieval Models

Let $f_q : \mathcal{Q} \to \mathbb{R}^d$ and $f_d : \mathcal{D} \to \mathbb{R}^d$ denote the query and document encoders respectively, where d is the dimension of the dense embedding space. The relevance score between a query q and a document d is computed via a similarity function $\text{sim}(f_q(q), f_d(d))$, which is often instantiated as the dot product or cosine similarity.

$$\text{sim}(q, d) = \langle f_q(q), f_d(d) \rangle \tag{2}$$

The retrieval distribution is then defined via a softmax over the similarity scores:

$$P(d \mid q) = \frac{\exp(\text{sim}(q, d)/\tau)}{\sum_{d' \in \mathcal{D}} \exp(\text{sim}(q, d')/\tau)} \tag{3}$$

where τ is a temperature hyperparameter controlling the sharpness of the distribution [9]. In practice, computing this softmax over the entire corpus $\mathcal{D}$ is computationally infeasible for large N, so Approximate Nearest Neighbor (ANN) search methods are employed to efficiently retrieve the top-k documents $\{d_1^*, \ldots, d_k^*\}$. Thus, the retrieval becomes:

 doi:10.20944/preprints202508.1211.v1

$$P(d \mid q) \approx \frac{\exp(\text{sim}(q,d)/\tau)}{\sum_{i=1}^{k} \exp(\text{sim}(q,d_i^*)/\tau)}, \quad d \in \{d_1^*, \ldots, d_k^*\} \tag{4}$$

2.3. Generative Modeling

Given a retrieved set of passages $D_q = \{d_1^*, \ldots, d_k^*\}$, the generator models the output distribution as:

$$P(y \mid q, D_q) = \text{LM}(y \mid [q; D_q]) \tag{5}$$

where $[q; D_q]$ denotes the concatenation of the query and the retrieved documents, and $\text{LM}(\cdot)$ represents a large pre-trained autoregressive language model (e.g., GPT, T5). The generator produces output tokens $y = (y_1, y_2, \ldots, y_T)$ sequentially, according to:

$$P(y \mid q, D_q) = \prod_{t=1}^{T} P(y_t \mid y_{<t}, q, D_q) \tag{6}$$

Several strategies exist to fuse the retrieved context D_q into the generation process, including:

- **Early Fusion:** Prepend D_q to the input prompt.
- **Late Fusion:** Combine outputs from multiple generations each conditioned on a single d_i^*.
- **Hierarchical Fusion:** Use attention mechanisms to dynamically weight the influence of different documents during decoding [10].

2.4. Training Objectives

The optimization of RAG systems can be conducted in multiple paradigms, including supervised fine-tuning, self-supervised learning, and reinforcement learning. A common supervised training objective is the marginal likelihood over retrieved documents:

$$\mathcal{L}_{\text{MLE}} = -\log \sum_{d \in D_q} P(y \mid q, d) P(d \mid q) \tag{7}$$

Since direct marginalization over D_q is computationally expensive, an evidence lower bound (ELBO) is typically maximized:

$$\mathcal{L}_{\text{ELBO}} = -\sum_{d \in D_q} P(d \mid q) \log P(y \mid q, d) \tag{8}$$

Alternatively, when ground-truth relevant documents are available, the retriever can be trained using contrastive learning. Given a positive document d^+ and a set of negatives $\{d_1^-, \ldots, d_m^-\}$, the contrastive loss is:

$$\mathcal{L}_{\text{contrast}} = -\log \frac{\exp(\text{sim}(q, d^+))}{\exp(\text{sim}(q, d^+)) + \sum_{j=1}^{m} \exp(\text{sim}(q, d_j^-))} \tag{9}$$

2.5. End-to-End Joint Training

Joint optimization of both retrieval and generation modules allows the two components to co-adapt, aligning the retriever to retrieve documents that are most useful for the downstream generation. One approach is to propagate gradients through the retrieved documents by treating $P(d \mid q)$ as a continuous variable, leading to:

$$\nabla \mathcal{L}_{\text{joint}} = \sum_{d \in D_q} \nabla P(d \mid q) \cdot \log P(y \mid q, d) + P(d \mid q) \cdot \nabla \log P(y \mid q, d) \tag{10}$$

In practice, the discrete nature of document selection poses challenges for backpropagation. Differentiable approximations such as Gumbel-Softmax or policy gradient-based reinforcement learning (e.g., REINFORCE) are commonly applied to facilitate joint optimization [11].

 doi:10.20944/preprints202508.1211.v1

2.6. Theoretical Considerations

From an information-theoretic perspective, RAG can be viewed as a mechanism to increase the mutual information between the query q and the generated output y by leveraging an auxiliary variable d. Formally:

$$I(q; y) \leq I(q; d) + I(d; y \mid q) \tag{11}$$

Thus, efficient and informative retrieval enhances the upper bound of the mutual information between input and output, leading to more relevant generations. Additionally, generalization bounds for RAG can be derived using PAC-Bayes analysis, where the generalization error depends on the complexity of the retriever and the generator, as well as their alignment. Regularizing the retriever to minimize overfitting (e.g., through entropy penalties on $P(d \mid q)$) helps ensure stable downstream performance.

2.7. Summary

This section formalized the RAG problem using probabilistic and information-theoretic notation, described the individual retrieval and generation mechanisms, and articulated the optimization objectives underpinning efficient training [12]. This mathematical foundation enables systematic exploration of trade-offs in retrieval quality, computational cost, model generalization, and downstream accuracy. In the following sections, we delve into specific classes of retrievers and generators, discuss efficiency improvements in large-scale settings, and analyze empirical performance across various RAG implementations [13].

3. Taxonomy of Efficient RAG Architectures

The design space of Retrieval-Augmented Generation (RAG) systems is highly modular, comprising diverse architectures that differ along multiple dimensions, such as the type of retriever used, the generator architecture, the mode of fusion between retrieved content and queries, and the training paradigms applied. To systematize the discussion and enable a structured comparison of methods, we propose a comprehensive taxonomy for efficient RAG systems. In this section, we elaborate on each axis of the taxonomy in detail and introduce a comparative table that summarizes key RAG models and their architectural and efficiency characteristics [14].

3.1. Taxonomy Dimensions

We define the following orthogonal dimensions to classify RAG systems:

- **Retriever Type:** Indicates whether the retriever is sparse (e.g., BM25), dense (e.g., dual-encoder), hybrid, or learned end-to-end.
- **Retriever Index:** Describes the index type used for efficient lookup (e.g., inverted index, FAISS-based ANN, HNSW).
- **Fusion Strategy:** Specifies how retrieved documents are integrated into the generation process—early fusion, late fusion, or attention-based fusion.
- **Generator Type:** Identifies the underlying architecture of the generator (e.g., GPT, BART, T5).
- **Training Paradigm:** Denotes how the system is trained—modular (separate retriever and generator training), jointly (end-to-end), or via reinforcement learning.
- **Context Handling:** Refers to how the model deals with multiple retrieved contexts—concatenation, reranking, or hierarchical modeling.
- **Efficiency Optimizations:** Includes caching, document compression, distillation, or retrieval-time pruning techniques.

 doi:10.20944/preprints202508.1211.v1

3.2. Comparison Table

Table 1. Comparison of representative RAG models across architectural and efficiency dimensions.

Model	Retriever Type	Index	Fusion	Generator	Training	Context Handling	Efficiency Techniques
REALM [?]	Dense (Dual Encoder)	FAISS	Early Fusion	BERT + MLM Head	Joint	Top-k Selection	Index Pretraining
RAG [?]	Dense + Sparse	IVF-Flat	Late Fusion (Marginalization)	BART	Modular	Reranking + Marginal	Fixed Top-k + Cache
FiD [?]	Dense (DPR)	Flat Index	Early Fusion (Encoder-level)	T5	Modular	Input Concatenation	Passage Dropout
REPLUG [?]	Dense	Flat + Cache	Attention-based Fusion	Decoder-only LLM (OPT)	Plug-and-Play	Sparse Top-k Filtering	Cache Distillation
ColBERT-QA [?]	Late Interaction (ColBERT)	ANN + Inverted Lists	Late Fusion + Scoring	BART	Modular	Linear Scoring over Candidates	Index Pruning
RETRO [?]	Dense + Exact Match	Chunk Index + Hash Map	Contextual Fusion (Cross-Attn)	Transformer Decoder	Modular	Chunked Fusion	Chunk Compression
Atlas [?]	Dense + Learned	FAISS HNSW	Early Fusion	T5	Joint (Retriever + Generator)	Full Joint Optimization	Retriever Fine-tuning
GNN-RAG [?]	Dense + Graph-based	HNSW + Graph Traversal	Attention Fusion + Graph Filtering	BART	Joint (Graph Encoder)	Relevance Propagation	Node Pruning
Dr. Retriever [?]	Dense + Memory-Augmented	Flat Memory	Early Fusion	GPT-2	Modular	Dialogue Turn-Aware	Memory Compression
Jina-RAG [?]	Hybrid (Sparse + Dense)	Jina Index (Hybrid ANN)	Early Fusion + Rerank	Encoder-Decoder	Modular	Adaptive Retrieval Depth	Query-aware Filtering

3.3. Analysis and Observations

From the comparative table, several trends emerge in the design of efficient RAG architectures:

Retriever Type and Indexing

Early RAG systems such as REALM and DPR relied primarily on dense retrievers with vector search indexes (e.g., FAISS). Recent systems adopt hybrid or even late-interaction retrievers (e.g., ColBERT) to strike a balance between efficiency and accuracy. Graph-enhanced retrieval (e.g., GNN-RAG) and cache-based methods (e.g., REPLUG) further optimize lookup performance [15].

Fusion Strategies

Simple early fusion techniques dominate among encoder-decoder architectures (e.g., FiD, Atlas), whereas decoder-only systems (e.g., RETRO) employ attention-based fusion with specialized memory integration. Late fusion marginalization, as used in RAG, is suitable for uncertainty-aware scenarios [16].

Training Paradigms

Modular training is still the dominant approach, as it decouples retriever and generator complexity [17]. However, joint training strategies have gained traction in end-to-end learning settings (e.g., Atlas), especially when retrieval errors significantly affect generation fidelity.

Efficiency Techniques

To ensure scalability, models incorporate various efficiency optimizations. For example, REPLUG uses retrieval cache distillation to reduce index search time, while FiD and RETRO apply document truncation and chunk compression respectively [18]. Top-k sampling, attention pruning, and retrieval frequency filtering are also prevalent [19].

 doi:10.20944/preprints202508.1211.v1

3.4. Concluding Remarks

This taxonomy underscores the breadth and diversity of RAG system design and highlights key trade-offs between performance, interpretability, and efficiency. Different application domains and system constraints—such as latency, throughput, and memory footprint—often dictate distinct architectural choices. In subsequent sections, we delve into these components individually, offering deeper insight into retrieval methods, scalable index structures, context integration strategies, and generation adaptations for real-time RAG deployment.

4. Efficient Retrieval Methods

Retrieval plays a central role in the Retrieval-Augmented Generation (RAG) paradigm. The effectiveness, scalability, and latency of the entire RAG pipeline are fundamentally influenced by the retrieval subsystem's quality and efficiency. This section delves into the diverse strategies and optimization techniques for implementing efficient retrieval in large-scale RAG systems. We organize the discussion around sparse, dense, hybrid, and learned retrieval models, followed by a treatment of indexing strategies, approximate nearest neighbor (ANN) search, and advanced retrieval-time optimizations [20].

4.1. Sparse Retrieval

Sparse retrieval methods rely on lexical overlap between the query and documents [21]. They are computationally inexpensive, interpretable, and easy to scale using inverted indexes. The classic approach is BM25, which assigns relevance scores based on term frequency, inverse document frequency, and document length normalization:

$$\text{BM25}(q,d) = \sum_{t \in q} \text{IDF}(t) \cdot \frac{f(t,d) \cdot (k_1 + 1)}{f(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})} \tag{12}$$

Sparse retrievers are effective in domains where precise keyword matching is required, such as legal documents, biomedical literature, or source code retrieval. However, they struggle with semantic matching, synonymy, and paraphrasing.

4.2. Dense Retrieval

Dense retrieval employs neural encoders to embed queries and documents into a shared continuous vector space. This allows semantic similarity matching using inner product or cosine similarity:

$$\text{sim}(q,d) = \langle f_q(q), f_d(d) \rangle \tag{13}$$

Notable dense retrieval models include:

- **DPR (Dense Passage Retrieval):** Uses BERT-based dual encoders trained with contrastive loss.
- **ANCE:** Applies hard negative mining and late-stage retraining [22].
- **ColBERT:** Utilizes late interaction via token-level embeddings and maximum-similarity aggregation.

Dense retrieval allows for better generalization, particularly in low-resource and multilingual settings. However, it typically requires more storage and computational resources due to the high-dimensional vector representations and ANN indexing.

4.3. Hybrid Retrieval

Hybrid retrievers combine sparse and dense signals, exploiting the precision of lexical matching and the generalization of semantic embedding. They are often implemented as:

$$\text{score}(q,d) = \lambda \cdot \text{BM25}(q,d) + (1 - \lambda) \cdot \text{sim}(q,d) \tag{14}$$

 doi:10.20944/preprints202508.1211.v1

Here, $\lambda \in [0,1]$ is a tunable interpolation weight. Hybrid methods outperform individual components in retrieval accuracy and robustness, especially in open-domain QA and multi-hop reasoning tasks.

4.4. Learned and Adaptive Retrieval

Recent efforts focus on end-to-end differentiable retrievers that can be fine-tuned jointly with the generator. These include:

- **Learned Dense Retrieval (LDR):** Retrieval scores are learned through a downstream generation loss.
- **GNN-based Retrieval:** Nodes in a document graph encode semantic and relational information; queries traverse the graph.
- **Retriever Distillation:** A smaller retriever mimics the behavior of a stronger teacher retriever (e.g., via KL divergence).

These models achieve high task-specific accuracy but often sacrifice interpretability and generality.

4.5. Indexing and ANN Search

Given the massive size of real-world corpora ($|\mathcal{D}| \gg 10^6$), exact nearest neighbor search becomes intractable. ANN methods approximate top-k retrieval using partitioning, quantization, or graph traversal. Popular libraries and techniques include:

- **FAISS:** Supports Product Quantization (PQ), Inverted File Index (IVF), and HNSW graphs.
- **ScaNN and NGT:** Optimized for high-speed ANN on GPU and CPU [23].
- **Hierarchical Navigable Small World (HNSW):** Graph-based method offering logarithmic search complexity.

Index tuning parameters—such as cluster size, number of probes, and compression depth—must be carefully optimized for trade-offs between latency, memory footprint, and recall.

4.6. Retrieval-Time Optimizations

Beyond encoder and index design, several strategies can be employed at retrieval time to improve efficiency:

Retrieval Caching

For frequently asked queries or sessions with high temporal locality, retrieval results can be cached and reused, dramatically reducing latency.

Dynamic Top-k Filtering

Instead of a fixed top-k, adaptive strategies rank candidates using lightweight heuristics before invoking full encoding and retrieval [24].

Query Pruning and Rewriting

Queries can be reformulated to remove stop words, identify entities, or use templates learned from past interactions to reduce retrieval complexity [25].

Compression and Quantization

Document vectors can be quantized (e.g., via PQ or binary hashing) to enable faster vector arithmetic and reduced memory usage.

4.7. Evaluation Metrics

Retrieval efficiency is typically evaluated on two axes:

- **Accuracy:** Measured by recall@k, precision@k, and mean reciprocal rank (MRR). High recall ensures useful documents are passed to the generator.

 doi:10.20944/preprints202508.1211.v1

- **Latency and Throughput:** Time per query and number of queries per second are essential for deployment.

The trade-off between recall and latency is central to efficient RAG design. System-level profiling is required to balance the time budget between retrieval and generation.

4.8. Summary

This section has dissected the retrieval component of RAG systems through a theoretical and empirical lens [26]. From sparse to dense to hybrid models, from static to learned and adaptive retrievers, the retrieval landscape is rich with optimization opportunities. Indexing techniques, ANN search, caching, and dynamic pruning further enhance scalability [27]. In the next section, we turn our attention to the generation component—how retrieved information is effectively and efficiently incorporated into the final output by modern foundation models.

5. Efficient Generation with Retrieved Contexts

In the Retrieval-Augmented Generation (RAG) pipeline, once relevant documents or passages are retrieved, the next challenge is incorporating this context into the generation process efficiently and effectively. This section provides a detailed examination of how foundation models integrate retrieved evidence to generate high-quality outputs, with a focus on strategies that improve computational efficiency, reduce latency, and maintain generation fidelity [28]. We explore architectural adaptations, context encoding techniques, attention mechanisms, parameter sharing, and training approaches used to make generation tractable in large-scale applications.

5.1. Context Fusion Strategies

Context fusion refers to the method by which retrieved documents $\{d_1, \ldots, d_k\}$ are incorporated into the language model to generate the output y. The main strategies include:

Early Fusion

All k documents are concatenated with the query q and processed jointly by the encoder. This is the approach used in Fusion-in-Decoder (FiD), where each passage is encoded independently and then fed into a decoder that attends to all passage encodings:

$$\text{FiD}(q, \{d_i\}) = \text{Decoder}(\text{Encoder}(q \oplus d_1), \ldots, \text{Encoder}(q \oplus d_k)) \tag{15}$$

Early fusion yields high accuracy due to deep integration but scales poorly with large k because the encoder and decoder attention complexity is quadratic in input length.

Late Fusion

Each document is processed independently, and their outputs are combined or marginalized [29]. For instance, in original RAG:

$$P(y|q) = \sum_{i=1}^{k} P(y|q, d_i) \cdot P(d_i|q) \tag{16}$$

This strategy reduces the computational load by avoiding attention across all passages simultaneously but may underutilize inter-document dependencies.

Retrieval-Aware Fusion (Attention-Based)

Recent models (e.g., RETRO, RePlug, Atlas) utilize cross-attention to retrieve-specific tokens or memory slots. The generator selectively attends to retrieved chunks using learned attention heads:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + \mathbf{m}\right)V \tag{17}$$

 doi:10.20944/preprints202508.1211.v1

where **m** may encode retrieval confidence or relevance weighting. This enables sparse, targeted integration and facilitates plug-and-play modularity.

5.2. Efficiency-Oriented Modifications

Several architectural and algorithmic modifications are designed to optimize generation speed and reduce computational overhead in RAG systems:

Truncation and Chunking

Only the top $t < k$ most relevant passages or the top l tokens per passage are retained [30]. Chunked passage representations are sometimes compressed into fixed-length vectors using pooling or learned adapters.

Sparse Attention and Routing

Attention is directed only to selected tokens or chunks to avoid full self-attention. Models like Longformer and BigBird use sparse attention patterns to reduce attention cost from $O(n^2)$ to $O(n)$ or $O(n \log n)$ [31].

Mixture-of-Experts (MoE)

Generation can be enhanced via MoE layers, where different sub-networks are specialized for different types of retrieval contexts:

$$\text{MoE}(x) = \sum_{i=1}^{M} g_i(x) \cdot E_i(x) \tag{18}$$

where g_i is a gating function and E_i is the expert network. This enables conditional computation and improves parameter efficiency.

Decoder Caching

Decoder states for previous tokens or document chunks are cached during autoregressive decoding. This reduces recomputation in left-to-right generation schemes.

Retrieval-Conditioned Prefixes

In prefix tuning or P-tuning, retrieved content is encoded into a small number of virtual tokens P_r that serve as a prefix to the input:

$$y \sim \text{LM}([P_r \oplus q]) \tag{19}$$

This allows control over generation with minimal additional computation and parameter tuning.

5.3. Parameter-Efficient Fine-Tuning

Full fine-tuning of foundation models is expensive and often unnecessary [32]. Efficient generation benefits from alternative tuning methods:

- **LoRA (Low-Rank Adaptation):** Updates low-rank matrices in attention layers while freezing the rest of the model.
- **Adapters:** Adds small bottleneck layers between transformer blocks that can be trained independently.
- **Prefix/Prompt Tuning:** Learns soft prompts or embeddings that condition the generation process.

These approaches reduce training time, memory usage, and support multi-task adaptation from a single frozen backbone.

5.4. Batching and Parallelism

In deployment scenarios, batching multiple queries and using parallel hardware accelerates RAG performance. FiD-style models can batch across retrieved documents as:

 doi:10.20944/preprints202508.1211.v1

$$\text{Batch} = \{(q_1, d_{1,1}), (q_1, d_{1,2}), \ldots, (q_B, d_{B,k})\} \tag{20}$$

where B is the batch size. Data and model parallelism (e.g., with tensor and pipeline parallelism) allow distributing encoder and decoder workloads across GPUs.

5.5. Evaluation of Generation Efficiency

To assess efficiency at the generation stage, several key metrics are used:

- **Latency per Token:** Measures the time to produce each output token.
- **Total Decode Time:** Captures end-to-end time for generating the complete response.
- **Throughput:** Number of generation requests per second in a production setting.
- **Context Utilization Ratio:** Fraction of retrieved documents that contribute non-trivially to the final output (e.g., via attention or copy) [33].

Balancing these metrics is crucial for latency-sensitive applications such as real-time assistants and retrieval-based code generation [34].

5.6. Summary

Efficient generation in RAG involves a complex interplay between architecture, context fusion, and system optimization. Techniques such as early and late fusion, sparse attention, prefix encoding, and parameter-efficient tuning all aim to reduce computational cost while maintaining or improving output quality. In the next section, we explore how RAG systems are trained end-to-end, including methods for aligning retrieval and generation, optimizing joint loss functions, and employing reinforcement learning to improve factuality and robustness.

6. Training Paradigms for Efficient RAG Systems

Training Retrieval-Augmented Generation (RAG) systems involves more than fine-tuning a language model with additional context. It requires synchronizing retrieval and generation components to jointly optimize end-task performance, while also maintaining or improving efficiency [35]. This section explores diverse training paradigms used in RAG: including supervised and self-supervised learning, contrastive training, end-to-end differentiable training, retrieval distillation, reinforcement learning, curriculum learning, and memory-efficient strategies.

6.1. Two-Stage vs. End-to-End Training

Training RAG systems generally follows two major paradigms:

Two-Stage Training

In this approach, the retriever and generator are trained separately. The retriever is first optimized using a retrieval-specific objective (e.g., contrastive loss, in-batch negatives), and then the generator is trained using the top-k retrieved documents:

$$\mathcal{L}_{\text{retriever}} = -\log \frac{\exp(\text{sim}(q, d^+))}{\sum_{d \in \mathcal{N}(q)} \exp(\text{sim}(q, d))} \tag{21}$$

$$\mathcal{L}_{\text{generator}} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, q, \{d_1, \ldots, d_k\}) \tag{22}$$

This separation improves modularity and scalability but may lead to suboptimal end-task performance due to retrieval-generation mismatch.

 doi:10.20944/preprints202508.1211.v1

End-to-End Training

Retriever and generator are optimized jointly by backpropagating the generation loss through the retriever. In the simplest case, retrieval scores are softmaxed to create a differentiable distribution over documents:

$$\mathcal{L}_{\text{joint}} = \mathbb{E}_{d \sim P(d|q)}[-\log P(y \mid q, d)] \tag{23}$$

Differentiability requires soft retrieval or gradient approximation (e.g., Gumbel-softmax, REINFORCE), which introduces computational overhead [36].

6.2. Contrastive and Supervised Retrieval Objectives

Efficient retrievers are trained using objectives that enforce semantic closeness between matched query-document pairs while separating negatives. The most common losses include:

In-Batch Negative Contrastive Loss

Each query in a batch treats other batch examples' documents as negatives:

$$\mathcal{L}_{\text{contrast}} = -\log \frac{\exp(\langle f_q(q_i), f_d(d_i) \rangle)}{\sum_{j=1}^{B} \exp(\langle f_q(q_i), f_d(d_j) \rangle)} \tag{24}$$

Triplet Loss

Based on hard or semi-hard negatives:

$$\mathcal{L}_{\text{triplet}} = \max(0, \text{sim}(q, d^-) - \text{sim}(q, d^+) + \alpha) \tag{25}$$

Label Supervision

If gold documents or references exist (e.g., in QA or summarization), supervised signals can be used to prioritize documents that maximize answer coverage.

6.3. Generator Fine-Tuning Objectives

Generators in RAG systems are typically fine-tuned using maximum likelihood estimation (MLE), but several advanced objectives are used to enhance faithfulness and fluency:

Sequence Likelihood

Standard objective across autoregressive models:

$$\mathcal{L}_{\text{MLE}} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, q, \{d_i\}) \tag{26}$$

Denoising and Masked Objectives

Inspired by T5 and BART pretraining; improves robustness when documents are partially corrupted.

Coverage and Saliency Regularization

Penalizes generation that ignores the retrieved context or hallucinates beyond it.

Multi-Task Learning

Combines generation with auxiliary tasks such as relevance prediction, fact checking, or summarization [37].

6.4. Reinforcement Learning (RL) for RAG

RL is used to optimize non-differentiable metrics like factuality, fluency, or user satisfaction. Policies are often trained with REINFORCE or actor-critic methods.

 doi:10.20944/preprints202508.1211.v1

Reward Functions

May include ROUGE/BLEU (fluency), F1/EM (QA), knowledge-grounding scores (e.g., knowledge F1), or learned reward models trained on human preferences.

$$\mathcal{L}_{\text{RL}} = -\mathbb{E}_{y \sim \pi_\theta}[R(y)] \tag{27}$$

Retrieval Policy Learning

RL can also be used to learn a retrieval policy $\pi(d \mid q)$ that selects documents maximizing downstream generation reward [38].

6.5. Distillation-Based Training

Distillation is used to train smaller or more efficient RAG models by mimicking larger models:

Retriever Distillation

Match the retriever scores or document rankings of a teacher:

$$\mathcal{L}_{\text{KD}}^{\text{retriever}} = \text{KL}(P_T(d \mid q) \| P_S(d \mid q)) \tag{28}$$

Generator Distillation:

Transfer token distributions or intermediate representations from a teacher model:

$$\mathcal{L}_{\text{KD}}^{\text{generator}} = \sum_t \text{KL}(P_T(y_t) \| P_S(y_t)) \tag{29}$$

6.6. Curriculum and Progressive Training

Curriculum learning helps reduce training complexity by introducing tasks in increasing order of difficulty:

- Start with synthetic or noisy retrievals and progress to gold documents.
- Use short, high-confidence queries before scaling to long-form or ambiguous ones.
- Gradually increase the number of retrieved documents during training.

6.7. Memory and Compute Efficient Training

Training large-scale RAG systems with long context is memory-intensive [39]. Efficiency strategies include:

- **Gradient Checkpointing:** Saves memory by recomputing intermediate activations during backpropagation.
- **Mixed Precision Training:** Reduces memory and improves throughput using FP16 or BF16 arithmetic.
- **Offline Retrieval:** Freeze and precompute document embeddings for faster training.
- **Retrieval Caching:** Reuse document retrievals across epochs when retrieval changes slowly.

6.8. Summary

The training of RAG systems demands careful coordination between retrieval and generation components. While two-stage training offers modularity, end-to-end and RL-based strategies yield better integration and task-specific performance [40]. Efficient training objectives—including contrastive losses, distillation, and curriculum learning—enable rapid convergence while managing resource constraints. The next section examines benchmarks, datasets, and evaluation protocols that allow consistent comparison across different RAG systems.

 doi:10.20944/preprints202508.1211.v1

7. Benchmarks and Evaluation Protocols

A critical component in the development and assessment of Retrieval-Augmented Generation (RAG) systems is rigorous evaluation [41]. Given the dual nature of RAG—retrieval followed by generation—evaluation must assess both retrieval quality and generation fidelity, as well as their synergy [42]. This section provides a comprehensive overview of datasets, benchmark tasks, metrics, and experimental protocols used to evaluate RAG systems across domains including question answering (QA), summarization, fact verification, and long-form generation.

7.1. Evaluation Dimensions

The performance of RAG systems is generally decomposed into the following dimensions:

1. **Retrieval Quality:** Measures how well the retriever selects relevant documents.
2. **Generation Quality:** Evaluates the fluency, informativeness, and correctness of the output.
3. **Faithfulness:** Assesses whether the generation adheres strictly to the retrieved evidence [43].
4. **Efficiency:** Includes latency, memory usage, and throughput during inference and training.

Each of these dimensions is assessed using domain-specific and domain-agnostic metrics, discussed below [44].

7.2. Benchmark Datasets

RAG systems are evaluated on a wide variety of benchmarks [45]. A selection of widely used datasets across different tasks is summarized in Table 2.

Table 2. Common Benchmarks for RAG Evaluation

Task	Dataset	Description	Retriever Input	Generator Target
Open-Domain QA	NaturalQuestions	Real-world questions from Google search logs	Question text	Answer string(s)
	TriviaQA	Trivia-style questions with evidence documents	Question	Answer string
	HotpotQA	Multi-hop QA requiring reasoning across docs	Question	Multi-sentence answer
	WebQuestions	QA from search queries linked to Freebase	Question	Named entity or phrase
Fact Checking	FEVER	Fact verification against Wikipedia	Claim statement	Label: SUPPORTS/REFUTES/NOT ENOUGH INFO
	SciFact	Scientific claim verification	Scientific claim	Label + Rationale sentence
Summarization	QMSum	Meeting transcript summarization	Dialogue transcripts	Abstractive summary
	MultiNews	Multi-document news summarization	Related news articles	Summary paragraph
Knowledge-Intensive Tasks	KILT	Unified benchmark covering QA, fact checking, entity linking, etc [46].	Task-specific	Varies
Long-Form Generation	ELI5	Reddit-based QA with long-form answers	Open-ended question	Paragraph-length answer
	NarrativeQA	Story understanding and narrative response	Story + Question	Long answer

7.3. Retrieval Evaluation Metrics

Evaluating the retriever independently provides insight into how well it supports downstream generation:

- **Recall@k:** Fraction of queries for which the relevant document appears in top-k results [47].
- **Precision@k:** Fraction of top-k retrieved documents that are relevant.
- **Mean Reciprocal Rank (MRR):** Average inverse rank of the first relevant document.
- **nDCG@k:** Normalized Discounted Cumulative Gain for relevance-ranked documents.
- **Embedding Similarity:** Cosine similarity between query and gold document embeddings [48].

For QA tasks with gold documents, Recall@k and MRR are especially popular.

 doi:10.20944/preprints202508.1211.v1

7.4. Generation Evaluation Metrics

To evaluate the generator, several automatic and human-centric metrics are used:

- **Exact Match (EM):** Binary match between predicted and ground-truth answer.
- **F1 Score:** Token overlap between predicted and true answer [49].
- **BLEU / ROUGE / METEOR:** N-gram based precision and recall measures.
- **BERTScore:** Semantic similarity via BERT embeddings [50].
- **Knowledge F1:** Measures whether generation correctly uses information from retrieved evidence.
- **Faithfulness / Hallucination Rate:** Measures how often generation includes unsupported claims.

7.5. Joint Retrieval-Generation Evaluation

Some tasks and metrics assess the joint performance of the RAG pipeline:

- **Answer Recall vs. Evidence Recall:** Measures how much answer correctness depends on evidence quality.
- **Precision-Recall Curves:** Derived from retrieval score thresholds that lead to successful generation.
- **Oracle vs. Retrieved Evaluation:** Compares performance using gold vs [51]. retrieved documents to assess retrieval bottlenecks [52].

7.6. Human Evaluation Protocols

Human studies are indispensable for evaluating faithfulness and usefulness. Typical criteria include:

- **Factual Accuracy:** Does the output contain any hallucinated or incorrect information?
- **Relevance:** Does the generation appropriately answer the query or summarize the source [53]?
- **Fluency and Readability:** Is the output grammatically sound and stylistically coherent [54]?
- **Usefulness:** Does the answer provide value in the context of the user's query [55]?

Crowd-sourcing platforms like Amazon Mechanical Turk and domain-expert annotations (e.g., for biomedical QA) are commonly used.

7.7. Efficiency Metrics

For practical deployment, RAG systems are also evaluated on:

- **Latency:** Total time from query input to generation output.
- **Retrieval Cost:** Time and resources required to retrieve documents [56].
- **Memory Footprint:** Peak GPU/CPU memory usage during inference.
- **Throughput:** Number of queries handled per second or per GPU hour.

These metrics are especially important for scaling RAG to enterprise search, assistants, and production QA systems [57].

7.8. Reproducibility and Reporting Practices

Due to the complexity of RAG pipelines, clear reporting is critical. Best practices include:

- Reporting performance with and without gold passages.
- Varying k (number of retrieved documents) to evaluate sensitivity.
- Isolating retrieval vs. generation contributions.
- Reporting both automatic and human evaluation metrics.
- Providing full code and data for replication [58].

7.9. Summary

Evaluation of RAG systems requires a holistic approach, combining traditional NLP metrics with retrieval-specific and human-centric evaluation [59]. Benchmark datasets spanning QA, summarization, and verification enable consistent comparisons, while metrics such as recall, F1, BERTScore, and hallucination rate provide fine-grained insights. Efficient RAG systems must also meet latency and

 doi:10.20944/preprints202508.1211.v1

throughput standards. In the following section, we examine recent trends and open challenges in building scalable, robust, and trustworthy RAG systems.

8. Challenges and Open Research Problems in Efficient RAG

While Retrieval-Augmented Generation (RAG) systems have demonstrated impressive capabilities across a wide range of NLP tasks, many core challenges remain unresolved—particularly when optimizing for both performance and efficiency. In this section, we outline key limitations in current RAG paradigms and identify open research questions that must be addressed to enable scalable, robust, and trustworthy RAG systems in real-world applications.

8.1. Retrieval-Generation Alignment Gap

One of the central challenges in RAG is the *retrieval-generation mismatch*: the retriever is typically trained to retrieve documents similar to the query, but not necessarily those that maximize generation utility. This misalignment often results in the generator producing incorrect or hallucinated outputs even when retrieved documents are topically relevant [60].

Open Problem

How can we optimize retrievers for *generation-aware relevance*, such that retrieved documents directly support the most likely correct output [61]?

Promising Directions

- Learn reward signals from generation loss gradients to guide retriever updates [62].
- Use reinforcement learning to jointly optimize retriever policies based on generation outcomes.
- Apply contrastive learning between relevant and spurious contexts with respect to output quality.

8.2. Faithfulness and Hallucination Control

Hallucination—when models generate unsupported or incorrect claims—remains a major barrier to deploying RAG systems in high-stakes settings like healthcare, law, or finance. While retrieval adds grounding, it does not guarantee factual consistency between context and output.

Open Problem

How can RAG systems be reliably constrained to generate only evidence-supported content?

Challenges

- Lack of fine-grained supervision linking output spans to document provenance.
- Absence of high-quality datasets for hallucination detection and control [63].
- Difficulty in balancing fluency and factuality [64].

Research Directions

- Incorporate faithfulness metrics (e.g., entailment models, attribution graphs) into loss functions [65].
- Develop generation architectures that condition more tightly on context, such as token-level attention masking or reranking [66].
- Use fact-checking modules as post-generation validators [67].

8.3. Scalability and Latency

Efficiency remains a critical challenge, especially when scaling RAG to web-scale retrieval corpora or deploying in low-latency environments such as conversational agents [68].

Open Problem

How can we design RAG architectures that maintain performance while reducing retrieval and inference costs?

 doi:10.20944/preprints202508.1211.v1

Key Bottlenecks

- Dense retrievers require large vector indexes and complex similarity search mechanisms.
- Long document inputs increase memory and computation cost for generators [69].
- Joint retriever-generator training increases gradient memory footprint.

Future Work

- Explore sparse or hybrid retrieval techniques with efficient indexes (e.g., ColBERT, SPLADE).
- Develop lightweight generator architectures with hierarchical or segment-aware encoding [70].
- Compress retriever and generator models via quantization, pruning, or knowledge distillation.

8.4. Context Selection and Redundancy

Many RAG systems rely on retrieving a fixed number k of top documents, which may introduce redundancy, noise, or conflicting information, especially when k is large.

Open Problem

How can systems dynamically select the optimal number and diversity of context documents [71]?

Research Opportunities

- Use uncertainty-aware retrieval strategies that balance relevance and novelty.
- Implement context re-ranking or clustering to improve information diversity [72].
- Allow generators to attend adaptively over documents rather than fixed concatenations.

8.5. Multimodal and Cross-Lingual RAG

Extending RAG beyond monolingual, text-only settings is increasingly relevant as applications demand multimodal and multilingual capabilities [73].

Open Problems

- How can RAG incorporate images, tables, or structured data as retrieval contexts?
- How can we train RAG systems for low-resource languages without parallel corpora?

Research Directions

- Develop shared embedding spaces for cross-modal or cross-lingual retrieval [74].
- Use translation-based retrieval or multilingual pretrained retrievers (e.g., mDPR, LaBSE).
- Design modality-aware generators that fuse information from heterogeneous sources.

8.6. Evaluation Challenges

Evaluation remains an open problem due to limitations in current metrics, which often fail to capture semantic fidelity, nuance, or downstream utility.

Key Limitations

- N-gram overlap metrics (e.g., ROUGE, BLEU) penalize valid but lexically dissimilar answers [75].
- Automatic faithfulness metrics are brittle and domain-specific.
- Human evaluation is expensive and inconsistent across studies [76].

Research Directions

- Develop learned reward models trained on large-scale human preference data.
- Use structured, extractive rationales for alignment between output and evidence [77].
- Standardize task-specific protocols for faithfulness and efficiency benchmarking [78].

 doi:10.20944/preprints202508.1211.v1

8.7. Adaptability and Lifelong Learning

Many RAG systems are static: once trained, their knowledge remains fixed [79]. Yet, real-world use cases require systems that adapt to new information, user behavior, or domain shifts.

Open Problem

How can RAG systems be made continuously adaptable without catastrophic forgetting or retraining?

Emerging Solutions

- Continually update the retrieval corpus and document embeddings without full retraining [80].
- Use episodic memory or retrieval-based meta-learning to adapt quickly to new tasks.
- Enable user feedback loops that influence retrieval and generation in real time [81].

8.8. Security, Privacy, and Bias

RAG systems may unintentionally retrieve or generate sensitive, biased, or misleading content [82]. Retrieval also introduces new attack vectors such as adversarial documents.

Risks

- Memorization of PII (Personally Identifiable Information) in generation.
- Exposure to harmful, toxic, or disinformation content through retrieval [83].
- Amplification of biases due to biased corpora or retrieval models [84].

Research Challenges

- Design privacy-preserving retrieval mechanisms (e.g., using differential privacy).
- Detect and filter harmful content during retrieval or post-generation.
- Mitigate bias via dataset balancing, fairness objectives, or counterfactual generation.

8.9. Summary

The RAG paradigm opens new avenues for efficient and controllable generation, but also raises a suite of technical and ethical challenges. These range from retrieval-generation alignment, faithfulness enforcement, and scalability, to the broader concerns of evaluation, adaptability, and safety. Addressing these challenges will require innovation across information retrieval, machine learning, human-computer interaction, and AI ethics. The next section reviews promising future directions and ongoing developments that aim to address these limitations.

9. Future Directions and Opportunities

Retrieval-Augmented Generation (RAG) represents a rapidly advancing frontier in natural language processing and foundation model research. As RAG systems evolve to meet the demands of increasingly complex, real-world tasks, a variety of future directions offer promise in making these systems more efficient, scalable, interpretable, and robust [85]. This section explores key avenues for advancing RAG beyond its current limitations, focusing on theoretical insights, architectural innovations, system design, and practical deployment strategies.

9.1. Unified End-to-End Optimization

Most current RAG pipelines consist of separately trained retrievers and generators, leading to potential suboptimal interactions between retrieval and generation modules [86]. Future research can focus on developing unified training objectives and architectures that allow for true end-to-end optimization.

Opportunities

- Design differentiable retrievers using approximations of discrete retrieval (e.g., Gumbel-softmax, REINFORCE, in-batch negatives).

 doi:10.20944/preprints202508.1211.v1

- Co-train retriever and generator with shared encoder backbones or dual encoders, enabling information flow between retrieval and generation stages [87].
- Investigate multitask and multi-objective frameworks combining retrieval, generation, and auxiliary tasks such as question decomposition or paraphrase generation [88].

9.2. Context-Aware and Query-Adaptive Retrieval

Current retrievers typically treat all queries uniformly and return a fixed number of documents. Future systems can improve efficiency and effectiveness by adapting retrieval strategies to the specific information needs of each query.

Ideas to Explore

- Develop *query-aware context policies* that determine how many documents to retrieve, based on uncertainty, ambiguity, or query intent [89].
- Implement *retrieval cascades*, where cheap shallow retrieval (e.g., BM25) is followed by dense reranking only when needed.
- Explore curriculum-based retrieval, where retrieval behavior evolves with the generator's learning phase.

9.3. Integration with Structured and Symbolic Knowledge

RAG systems currently rely heavily on unstructured natural language corpora. A promising direction is the integration of structured sources—such as knowledge graphs, tables, or code—into the retrieval process, enhancing factual precision and logical reasoning capabilities [90].

Challenges and Goals

- Design hybrid retrievers capable of jointly querying both free-text and structured knowledge.
- Encode graph-based retrieval results (e.g., paths in a knowledge graph) into textual contexts that preserve relational semantics.
- Use symbolic reasoning over retrieved facts to improve multi-hop and commonsense inference.

9.4. Multi-Hop and Compositional Reasoning

Future RAG systems should be capable of retrieving and reasoning over multiple documents in a compositional manner to answer complex, multi-step queries [91].

Promising Approaches

- Sequential retrieval pipelines where intermediate answers inform subsequent retrieval steps.
- Graph-based modeling of evidence chains, enabling backward and forward chaining reasoning [92].
- Integrate RAG with neuro-symbolic methods to trace and verify reasoning steps explicitly.

9.5. Personalized and Continual Retrieval-Augmented Learning

Personalized assistants and domain-adaptive systems need to reflect evolving user knowledge and preferences [93]. Future RAG models can benefit from memory-augmented, adaptive systems that learn over time.

Future Directions

- Design long-term memory components that store user interactions, personal documents, or task-specific knowledge.
- Implement continual learning frameworks that update document encodings and retriever parameters without catastrophic forgetting [94].
- Explore federated or on-device learning techniques for privacy-aware personalization [95].

 doi:10.20944/preprints202508.1211.v1

9.6. Interpretable and Transparent RAG Systems

As RAG systems are increasingly applied in high-stakes domains, interpretability becomes essential. Future RAG architectures should offer users and developers insight into how outputs are derived from retrieved evidence [96].

Research Goals

- Provide traceability from generation output to specific retrieved passages [97].
- Develop explanation interfaces that highlight influential evidence spans and reasoning chains [98].
- Incorporate attribution mechanisms (e.g., attention maps, saliency methods) into training objectives to improve transparency.

9.7. RAG in Multimodal and Interactive Settings

Expanding RAG beyond static text environments to handle multimodal input (e.g., images, audio, video) and dynamic, interactive contexts is a critical step toward generalized intelligence.

Emerging Frontiers:

- Design retrievers for multimodal corpora, supporting inputs like images, tables, or time-series data.
- Create multimodal encoders that align vision, language, and audio embeddings into shared retrieval spaces.
- Enable dialog-centric RAG that incorporates user feedback and performs live context updates [99].

9.8. Evaluation Frameworks for Next-Gen RAG

As RAG systems diversify in application and architecture, evaluation protocols must evolve to support new desiderata: adaptiveness, generalization, safety, and interactivity [100].

Opportunities

- Develop benchmark suites that test end-to-end performance across multiple axes (e.g., factuality, novelty, response diversity).
- Introduce task-agnostic metrics that quantify retrieval-generation alignment or evidence sensitivity.
- Embrace simulation environments (e.g., embodied QA, virtual assistants) for real-time interactive evaluation.

9.9. Synergies with Agent Architectures

RAG can serve as the backbone for tool-using agents that retrieve, reason, and generate in a closed loop. Agentic systems using RAG can dynamically retrieve code snippets, papers, or database entries during decision making.

Examples and Opportunities

- Integrate RAG with planning modules, memory stores, and tool execution layers.
- Use RAG to build agents that learn retrieval heuristics via reinforcement learning.
- Enable recursive RAG pipelines where generations trigger further retrievals, enabling autonomous workflows.

9.10. Summary

The path forward for Retrieval-Augmented Generation spans deep theoretical, architectural, and practical territory. Future RAG systems are likely to be more integrated, dynamic, multimodal, and user-centric, supporting interactive applications and personalized experiences [101]. Achieving this vision requires not only better models but also new training strategies, evaluation paradigms, and responsible deployment practices. By addressing these future directions, the field can unlock the full potential of RAG as a foundational component of intelligent, grounded, and efficient AI systems [102].

 doi:10.20944/preprints202508.1211.v1

10. Conclusions

Retrieval-Augmented Generation (RAG) has emerged as a transformative paradigm for combining the strengths of parametric knowledge encoded within large-scale language models and non-parametric external memory via retrieval mechanisms. As foundational models continue to scale, the necessity for more interpretable, updatable, and efficient systems becomes paramount—RAG offers a compelling solution to this challenge.

This survey has provided a comprehensive and detailed exploration of the theoretical foundations, architectural variants, efficiency-oriented techniques, empirical benchmarks, and persistent challenges in the field of efficient RAG. We began by formalizing the RAG framework using rigorous mathematical notations, examining its probabilistic underpinnings, modular components, and variants. We then systematically cataloged a wide range of efficiency techniques spanning both the retrieval and generation subsystems, emphasizing trade-offs in latency, memory, throughput, and accuracy. Our comparative analysis highlighted the practical implications of these techniques across several benchmark datasets, tasks, and domains.

In addition to summarizing state-of-the-art practices, we identified multiple open research questions that represent significant barriers to deployment and innovation in RAG. These challenges include optimizing for retrieval-generation alignment, minimizing hallucination, improving generalization under domain shift, achieving robust multi-hop reasoning, and addressing safety, fairness, and privacy concerns.

To help guide future work, we outlined a series of promising research directions, such as unified retriever-generator training, adaptive and personalized retrieval policies, integration with structured knowledge, multimodal expansion, and synergy with agent-based systems. As the ecosystem around RAG continues to mature, we expect these areas to become increasingly important in both academic and industrial contexts.

RAG is not merely a stopgap solution for current limitations in language modeling but is likely to be a cornerstone in the evolution of grounded, trustworthy, and efficient AI systems. By effectively leveraging external knowledge retrieval while maintaining the fluency and flexibility of generative models, RAG enables new levels of performance and interpretability across a spectrum of AI applications—from open-domain question answering and scientific discovery to personalized assistants and decision support systems.

Ultimately, the continued progress of RAG will depend on advances in algorithm design, training methodologies, scalable infrastructures, and ethical deployment practices. We hope this survey serves as both a foundational reference and a forward-looking guide for researchers, engineers, and practitioners who aim to push the boundaries of Retrieval-Augmented Generation toward more capable, efficient, and aligned intelligent systems.

References

1. Cheng, X.; Gao, S.; Liu, L.; Zhao, D.; Yan, R. Neural machine translation with contrastive translation memories. *arXiv preprint arXiv:2212.03140* **2022**.
2. Xu, Z.; Gong, Y.; Zhou, Y.; Bao, Q.; Qian, W. Enhancing Kubernetes Automated Scheduling with Deep Learning and Reinforcement Techniques for Large-Scale Cloud Computing Optimization, 2024, [arXiv:cs.DC/2403.07905].
3. Cheng, D.; Huang, S.; Bi, J.; Zhan, Y.; Liu, J.; Wang, Y.; Sun, H.; Wei, F.; Deng, D.; Zhang, Q. UPRISE: Universal Prompt Retrieval for Improving Zero-Shot Evaluation. *arXiv preprint arXiv:2303.08518* **2023**.
4. Joshi, M.; Choi, E.; Weld, D.S.; Zettlemoyer, L. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension, 2017, [arXiv:cs.CL/1705.03551].
5. Jeong, S.; Baek, J.; Cho, S.; Hwang, S.J.; Park, J.C. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity, 2024, [arXiv:cs.CL/2403.14403].
6. Anderson, N.; Wilson, C.; Richardson, S.D. Lingua: Addressing Scenarios for Live Interpretation and Automatic Dubbing. In Proceedings of the Proceedings of the 15th Biennial Conference of the Association

 doi:10.20944/preprints202508.1211.v1

for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track); Campbell, J.; Larocca, S.; Marciano, J.; Savenkov, K.; Yanishevsky, A., Eds., Orlando, USA, 2022; pp. 202–209.
7. Ebner, S.; Xia, P.; Culkin, R.; Rawlins, K.; Durme, B.V. Multi-Sentence Argument Linking, 2020, [arXiv:cs.CL/1911.03766].
8. AG2AI Contributors. AG2: A GitHub Repository for Advanced Generative AI Research. https://github.com/ag2ai/ag2, 2025. Accessed: 2025-01-15.
9. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv* **2023**, arXiv:2307.09288.
10. An, Z.; Ding, X.; Fu, Y.C.; Chu, C.C.; Li, Y.; Du, W. Golden-Retriever: High-Fidelity Agentic Retrieval Augmented Generation for Industrial Knowledge Base, 2024, [arXiv:cs.IR/2408.00798].
11. Blog, Q. Agentic RAG: Combining RAG with Agents for Enhanced Information Retrieval. https://qdrant.tech/articles/agentic-rag/. Accessed: 2025-01-14.
12. Dai, Z.; Zhao, V.Y.; Ma, J.; Luan, Y.; Ni, J.; Lu, J.; Bakalov, A.; Guu, K.; Hall, K.B.; Chang, M.W. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755* **2022**.
13. Izacard, G.; Grave, E. Distilling knowledge from reader to retriever for question answering. *arXiv preprint arXiv:2012.04584* **2020**.
14. Li, X.; Zhao, R.; Chia, Y.K.; Ding, B.; Bing, L.; Joty, S.; Poria, S. Chain of Knowledge: A Framework for Grounding Large Language Models with Structured Knowledge Bases. *arXiv preprint arXiv:2305.13269* **2023**.
15. NVIDIA. Spectrum-X: End-to-End Networking for AI and High-Performance Computing. https://www.nvidia.com/en-us/networking/spectrumx/, 2025. Accessed: 2025-01-28.
16. Anantha, R.; Bethi, T.; Vodianik, D.; Chappidi, S. Context Tuning for Retrieval Augmented Generation. *arXiv preprint arXiv:2312.05708* **2023**.
17. Thakur, N.; Bonifacio, L.; Zhang, X.; Ogundepo, O.; Kamalloo, E.; Alfonso-Hermelo, D.; Li, X.; Liu, Q.; Chen, B.; Rezagholizadeh, M.; et al. "Knowing When You Don't Know": A Multilingual Relevance Assessment Dataset for Robust Retrieval-Augmented Generation, 2024, [arXiv:cs.CL/2312.11361].
18. Friel, R.; Belyi, M.; Sanyal, A. RAGBench: Explainable Benchmark for Retrieval-Augmented Generation Systems, 2024, [arXiv:cs.CL/2407.11005].
19. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
20. Ma, Y.; Cao, Y.; Hong, Y.; Sun, A. Large Language Model Is Not a Good Few-shot Information Extractor, but a Good Reranker for Hard Samples! *ArXiv* **2023**, *abs/2303.08559*.
21. Repository, L.D. Auto Insurance Claims Workflow using LlamaCloud. https://github.com/run-llama/llamacloud-demo/blob/main/examples/document_workflows/auto_insurance_claims/auto_insurance_claims.ipynb, 2025. Accessed: 2025-01-13.
22. Blog, N.D. Microsoft GraphRAG and Neo4j, 2024. Accessed: 2025-01-11.
23. Zhong, Z.; Lei, T.; Chen, D. Training language models with memory augmentation. *arXiv preprint arXiv:2205.12674* **2022**.
24. Ho, X.; Nguyen, A.K.D.; Sugawara, S.; Aizawa, A. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060* **2020**.
25. Nakano, R.; Hilton, J.; Balaji, S.; Wu, J.; Ouyang, L.; Kim, C.; Hesse, C.; Jain, S.; Kosaraju, V.; Saunders, W.; et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv* **2021**, arXiv:2112.09332.
26. Singh, A.; Ehtesham, A.; Kumar, S.; Gupta, G.K.; Khoei, T.T. Encouraging Responsible Use of Generative AI in Education: A Reward-Based Learning Approach. In Proceedings of the Artificial Intelligence in Education Technologies: New Development and Innovative Practices; Schlippe, T.; Cheng, E.C.K.; Wang, T., Eds., Singapore, 2025; pp. 404–413.
27. Zniyed, Y.; Nguyen, T.P.; et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems* **2024**.
28. Li, S.; Ji, H.; Han, J. Document-Level Event Argument Extraction by Conditional Generation, 2021, [arXiv:cs.CL/2104.05919].
29. Wang, S.; Xu, Y.; Fang, Y.; Liu, Y.; Sun, S.; Xu, R.; Zhu, C.; Zeng, M. Training Data is More Valuable than You Think: A Simple and Effective Method by Retrieving from Training Data. In Proceedings of the Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Muresan, S.; Nakov, P.; Villavicencio, A., Eds., Dublin, Ireland, 2022; pp. 3170–3179. https://doi.org/10.18653/v1/2022.acl-long.226.

 doi:10.20944/preprints202508.1211.v1

30. Husain, H.; Wu, H.H.; Gazit, T.; Allamanis, M.; Brockschmidt, M. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436* **2019**.
31. Tutorial, L.A.R. LangGraph Adaptive RAG: Adaptive Retrieval-Augmented Generation Tutorial. https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_adaptive_rag/. Accessed: 2025-01-14.
32. Narayan, S.; Cohen, S.B.; Lapata, M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *arXiv preprint arXiv:1808.08745* **2018**.
33. Cerny, T.; Abdelfattah, A.S.; Bushong, V.; Maruf, A.A.; Taibi, D. Microservice Architecture Reconstruction and Visualization Techniques: A Review, 2022, [arXiv:cs.SE/2207.02988].
34. Blog, N.D. Indexing Pipeline GraphRAG Image, 2024. Accessed: 2025-01-11.
35. Borgeaud, S.; Mensch, A.; Hoffmann, J.; Cai, T.; Rutherford, E.; Millican, K.; Van Den Driessche, G.B.; Lespiau, J.B.; Damoc, B.; Clark, A.; et al. Improving language models by retrieving from trillions of tokens. In Proceedings of the International conference on machine learning. PMLR, 2022, pp. 2206–2240.
36. Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* **2021**.
37. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761* **2023**.
38. Berchansky, M.; Izsak, P.; Caciularu, A.; Dagan, I.; Wasserblat, M. Optimizing Retrieval-augmented Reader Models via Token Elimination. *arXiv preprint arXiv:2310.13682* **2023**.
39. Hoshi, Y.; Miyashita, D.; Ng, Y.; Tatsuno, K.; Morioka, Y.; Torii, O.; Deguchi, J. RaLLe: A Framework for Developing and Evaluating Retrieval-Augmented Large Language Models. *arXiv* **2023**, arXiv:2308.10633.
40. Berchansky, M.; Izsak, P.; Caciularu, A.; Dagan, I.; Wasserblat, M. Optimizing retrieval-augmented reader models via token elimination. *arXiv preprint arXiv:2310.13682* **2023**.
41. Rajput, S.; Mehta, N.; Singh, A.; Keshavan, R.H.; Vu, T.; Heldt, L.; Hong, L.; Tay, Y.; Tran, V.Q.; Samost, J.; et al. Recommender Systems with Generative Retrieval. *arXiv preprint arXiv:2305.05065* **2023**.
42. Wang, X.; Chen, G.H.; Song, D.; Zhang, Z.; Chen, Z.; Xiao, Q.; Jiang, F.; Li, J.; Wan, X.; Wang, B.; et al. CMB: A Comprehensive Medical Benchmark in Chinese, 2024, [arXiv:cs.CL/2308.08833].
43. Gupta, G.K.; Singh, A.; Manikandan, S.V.; Ehtesham, A. Digital Diagnostics: The Potential of Large Language Models in Recognizing Symptoms of Common Illnesses. *AI* **2025**, *6*. https://doi.org/10.3390/ai6010013.
44. Shi, F.; Chen, X.; Misra, K.; Scales, N.; Dohan, D.; Chi, E.H.; Schärli, N.; Zhou, D. Large language models can be easily distracted by irrelevant context. In Proceedings of the International Conference on Machine Learning. PMLR, 2023, pp. 31210–31227.
45. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework. 2023, [arXiv:cs.AI/2308.08155].
46. Geva, M.; Khashabi, D.; Segal, E.; Khot, T.; Roth, D.; Berant, J. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies, 2021, [arXiv:cs.CL/2101.02235].
47. Fan, A.; Jernite, Y.; Perez, E.; Grangier, D.; Weston, J.; Auli, M. ELI5: Long form question answering. *arXiv preprint arXiv:1907.09190* **2019**.
48. Sciavolino, C.; Zhong, Z.; Lee, J.; Chen, D. Simple entity-centric questions challenge dense retrievers. *arXiv preprint arXiv:2109.08535* **2021**.
49. Rau, D.; Déjean, H.; Chirkova, N.; Formal, T.; Wang, S.; Nikoulina, V.; Clinchant, S. BERGEN: A Benchmarking Library for Retrieval-Augmented Generation, 2024, [arXiv:cs.CL/2407.01102].
50. Xia, M.; Huang, G.; Liu, L.; Shi, S. Graph based translation memory for neural machine translation. In Proceedings of the Proceedings of the AAAI conference on artificial intelligence, 2019, Vol. 33, pp. 7297–7304.
51. Geva, M.; Khashabi, D.; Segal, E.; Khot, T.; Roth, D.; Berant, J. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics* **2021**, *9*, 346–361.
52. Liu, Y.; Yavuz, S.; Meng, R.; Moorthy, M.; Joty, S.; Xiong, C.; Zhou, Y. Exploring the integration strategies of retriever and large language models. *arXiv preprint arXiv:2308.12574* **2023**.
53. Yang, A.; Nagrani, A.; Seo, P.H.; Miech, A.; Pont-Tuset, J.; Laptev, I.; Sivic, J.; Schmid, C. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 10714–10726.
54. Yan, S.Q.; Gu, J.C.; Zhu, Y.; Ling, Z.H. Corrective Retrieval Augmented Generation, 2024, [arXiv:cs.CL/2401.15884].

 doi:10.20944/preprints202508.1211.v1

55. Liu, N.F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; Liang, P. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172* **2023**.
56. Trivedi, H.; Balasubramanian, N.; Khot, T.; Sabharwal, A. MuSiQue: Multihop Questions via Single-hop Question Composition, 2022, [arXiv:cs.CL/2108.00573].
57. Packer, C.; Fang, V.; Patil, S.G.; Lin, K.; Wooders, S.; Gonzalez, J.E. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560* **2023**.
58. Singh, A. Exploring Language Models: A Comprehensive Survey and Analysis. In Proceedings of the 2023 International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE), 2023, pp. 1–4. https://doi.org/10.1109/RMKMATE59243.2023.10369423.
59. Lin, X.V.; Chen, X.; Chen, M.; Shi, W.; Lomeli, M.; James, R.; Rodriguez, P.; Kahn, J.; Szilvasy, G.; Lewis, M.; et al. RA-DIT: Retrieval-Augmented Dual Instruction Tuning. *arXiv preprint arXiv:2310.01352* **2023**.
60. Bajaj, P.; Campos, D.; Craswell, N.; Deng, L.; Gao, J.; Liu, X.; Majumder, R.; McNamara, A.; Mitra, B.; Nguyen, T.; et al. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset, 2018, [arXiv:cs.CL/1611.09268].
61. Jarvis, C.; Allard, J. A Survey of Techniques for Maximizing LLM Performance. https://community.openai.com/t/openai-dev-day-2023-breakout-sessions/505213#a-survey-of-techniques-for-maximizing-llm-performance-2, 2023.
62. Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; Gurevych, I. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663* **2021**.
63. Petroni, F.; Rocktäschel, T.; Lewis, P.; Bakhtin, A.; Wu, Y.; Miller, A.H.; Riedel, S. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066* **2019**.
64. Peng, B.; Zhu, Y.; Liu, Y.; Bo, X.; Shi, H.; Hong, C.; Zhang, Y.; Tang, S. Graph Retrieval-Augmented Generation: A Survey, 2024, [arXiv:cs.AI/2408.08921].
65. Mallen, A.; Asai, A.; Zhong, V.; Das, R.; Hajishirzi, H.; Khashabi, D. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511* **2022**.
66. Baek, J.; Aji, A.F.; Saffari, A. Knowledge-Augmented Language Model Prompting for Zero-Shot Knowledge Graph Question Answering. *arXiv preprint arXiv:2306.04136* **2023**.
67. Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300* **2020**.
68. Xu, P.; Ping, W.; Wu, X.; McAfee, L.; Zhu, C.; Liu, Z.; Subramanian, S.; Bakhturina, E.; Shoeybi, M.; Catanzaro, B. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025* **2023**.
69. Documentation, L. Agentic RAG Using Vertex AI. https://docs.llamaindex.ai/en/stable/examples/agent/agentic_rag_using_vertex_ai/. Accessed: 2025-01-14.
70. Zhao, P.; Zhang, H.; Yu, Q.; Wang, Z.; Geng, Y.; Fu, F.; Yang, L.; Zhang, W.; Jiang, J.; Cui, B. Retrieval-Augmented Generation for AI-Generated Content: A Survey, 2024, [arXiv:cs.CV/2402.19473].
71. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q.V.; Zhou, D.; et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* **2022**, *35*, 24824–24837.
72. Robertson, S.; Zaragoza, H.; et al. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* **2009**, *3*, 333–389.
73. Luo, Z.; Xu, C.; Zhao, P.; Geng, X.; Tao, C.; Ma, J.; Lin, Q.; Jiang, D. Augmented Large Language Models with Parametric Knowledge Guiding. *arXiv preprint arXiv:2305.04757* **2023**.
74. Microsoft. Semantic Kernel Overview, 2025. https://learn.microsoft.com/en-us/semantic-kernel/overview/. Accessed: February 2, 2025.
75. Xu, S.; Pang, L.; Shen, H.; Cheng, X.; Chua, T.S. Search-in-the-chain: Towards accurate, credible and traceable large language models for knowledgeintensive tasks. *CoRR, vol. abs/2304.14732* **2023**.
76. Yasunaga, M.; Aghajanyan, A.; Shi, W.; James, R.; Leskovec, J.; Liang, P.; Lewis, M.; Zettlemoyer, L.; Yih, W.t. Retrieval-augmented multimodal language modeling. *arXiv preprint arXiv:2211.12561* **2022**.
77. Wang, H.; Hu, M.; Deng, Y.; Wang, R.; Mi, F.; Wang, W.; Wang, Y.; Kwan, W.C.; King, I.; Wong, K.F. Large Language Models as Source Planner for Personalized Knowledge-grounded Dialogue. *arXiv preprint arXiv:2310.08840* **2023**.
78. Jiang, H.; Wu, Q.; Lin, C.Y.; Yang, Y.; Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736* **2023**.

 doi:10.20944/preprints202508.1211.v1

79. Zhao, W.X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. A Survey of Large Language Models, 2024, [arXiv:cs.CL/2303.18223].
80. Wen, T.H.; Gašić, M.; Mrkšić, N.; Rojas-Barahona, L.M.; Su, P.H.; Ultes, S.; Vandyke, D.; Young, S. Conditional Generation and Snapshot Learning in Neural Dialogue Systems. In Proceedings of the Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, Texas, 2016; pp. 2153–2162. https://doi.org/10.18653/v1/D16-1233.
81. Wang, H.; Huang, W.; Deng, Y.; Wang, R.; Wang, Z.; Wang, Y.; Mi, F.; Pan, J.Z.; Wong, K.F. UniMS-RAG: A Unified Multi-source Retrieval-Augmented Generation for Personalized Dialogue Systems. *arXiv preprint arXiv:2401.13256* **2024**.
82. Saha, S.; Junaed, J.A.; Saleki, M.; Sen Sharma, A.; Rifat, M.R.; Rahouti, M.; Ahmed, S.I.; Mohammed, N.; Amin, M.R. Vio-Lens: A Novel Dataset of Annotated Social Network Posts Leading to Different Forms of Communal Violence and its Evaluation. In Proceedings of the Proceedings of the First Workshop on Bangla Language Processing (BLP-2023); Alam, F.; Kar, S.; Chowdhury, S.A.; Sadeque, F.; Amin, R., Eds., Singapore, 2023; pp. 72–84. https://doi.org/10.18653/v1/2023.banglalp-1.9.
83. Saha, S.; Junaed, J.A.; Saleki, M.; Sharma, A.S.; Rifat, M.R.; Rahouti, M.; Ahmed, S.I.; Mohammed, N.; Amin, M.R. Vio-lens: A novel dataset of annotated social network posts leading to different forms of communal violence and its evaluation. In Proceedings of the Proceedings of the First Workshop on Bangla Language Processing (BLP-2023), 2023, pp. 72–84.
84. Li, J.; Li, D.; Savarese, S.; Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597* **2023**.
85. Merity, S.; Xiong, C.; Bradbury, J.; Socher, R. Pointer Sentinel Mixture Models, 2016, [arXiv:cs.CL/1609.07843].
86. Liu, X.; Lai, H.; Yu, H.; Xu, Y.; Zeng, A.; Du, Z.; Zhang, P.; Dong, Y.; Tang, J. WebGLM: Towards An Efficient Web-Enhanced Question Answering System with Human Preferences. *arXiv preprint arXiv:2306.07906* **2023**.
87. Blog, L. Agentic RAG with LlamaIndex. https://www.llamaindex.ai/blog/agentic-rag-with-llamaindex-2721b8a49ff6. Accessed: 2025-01-14.
88. Shi, T.; Li, L.; Lin, Z.; Yang, T.; Quan, X.; Wang, Q. Dual-Feedback Knowledge Retrieval for Task-Oriented Dialogue Systems. *arXiv preprint arXiv:2310.14528* **2023**.
89. Gottlob, G. Complexity results for nonmonotonic logics. *Journal of Logic and Computation* **1992**, *2*, 397–425.
90. Asai, A.; Wu, Z.; Wang, Y.; Sil, A.; Hajishirzi, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv preprint arXiv:2310.11511* **2023**.
91. Steinberger, R.; Pouliquen, B.; Widiger, A.; Ignat, C.; Erjavec, T.; Tufiş, D.; Varga, D. The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages. In Proceedings of the Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06); Calzolari, N.; Choukri, K.; Gangemi, A.; Maegaard, B.; Mariani, J.; Odijk, J.; Tapias, D., Eds., Genoa, Italy, 2006.
92. Tutorial, L.C. LangGraph CRAG: Contextualized Retrieval-Augmented Generation Tutorial. https://langchain-ai.github.io/langgraph/tutorials/rag/langgraph_crag/. Accessed: 2025-01-14.
93. Jiang, Z.; Xu, F.F.; Gao, L.; Sun, Z.; Liu, Q.; Dwivedi-Yu, J.; Yang, Y.; Callan, J.; Neubig, G. Active Retrieval Augmented Generation, 2023, [arXiv:cs.CL/2305.06983].
94. Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; Mittal, A. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In Proceedings of the Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers); Walker, M.; Ji, H.; Stent, A., Eds., New Orleans, Louisiana, 2018; pp. 809–819. https://doi.org/10.18653/v1/N18-1074.
95. Melz, E. Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation. *arXiv preprint arXiv:2311.04177* **2023**.
96. crewAI Inc.. crewAI: A GitHub Repository for AI Projects. https://github.com/crewAIInc/crewAI, 2025. Accessed: 2025-01-15.
97. Wang, H.; Hu, M.; Deng, Y.; Wang, R.; Mi, F.; Wang, W.; Wang, Y.; Kwan, W.C.; King, I.; Wong, K.F. Large Language Models as Source Planner for Personalized Knowledge-grounded Dialogue, 2023, [arXiv:cs.CL/2310.08840].
98. Guo, Z.; Cheng, S.; Wang, Y.; Li, P.; Liu, Y. Prompt-Guided Retrieval Augmentation for Non-Knowledge-Intensive Tasks. *arXiv preprint arXiv:2305.17653* **2023**.
99. Kim, G.; Kim, S.; Jeon, B.; Park, J.; Kang, J. Tree of Clarifications: Answering Ambiguous Questions with Retrieval-Augmented Large Language Models. *arXiv preprint arXiv:2310.14696* **2023**.

 doi:10.20944/preprints202508.1211.v1

100. Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; et al. Self-Refine: Iterative Refinement with Self-Feedback, 2023, [arXiv:cs.CL/2303.17651].
101. LangChain. LangSmith: The Ultimate Toolkit for Debugging and Monitoring LLM Applications. https://www.langchain.com/langsmith, 2025. Accessed: 2025-01-28.
102. Yang, S. Advanced RAG 01: Small-to-Big Retrieval. https://towardsdatascience.com/advanced-rag-01-small-to-big-retrieval-172181b396d4, 2023.