Fundamental Algorithms for Retrieval Augmented Generation: A Survey

Zihao Huang, Rui Mao*, *Member, IEEE*, Xiaobao Wu, Kai He, Xulang Zhang, Erik Cambria, *Fellow, IEEE*

***Abstract*—Large language models have achieved remarkable success in natural language generation, yet their reliance on static, pre-trained parameters limits their ability to provide accurate and up-to-date information. Retrieval-Augmented Generation (RAG) addresses this limitation by integrating external knowledge retrieval into the generation process, making it a powerful framework for knowledge-intensive tasks. As RAG systems evolve, researchers have investigated various technical approaches, ranging from differing architectures for retrievers and generators to complex combinations of the two. However, few surveys provide a unified view of RAG from a mathematical and algorithmic perspective. In this survey, we present a systematic analysis of RAG systems from the perspective of fundamental algorithms. We introduce representative algorithms with formal definitions, and examine how modern RAG systems select and combine these components in practice. We also summarize typical application domains and benchmark performance across tasks, showing the strengths and weaknesses of different algorithms. Our study provides a structured foundation for understanding the principles, implementations, and challenges of RAG.**

***Index Terms*—Retrieval Augmented Generation, Large Language Model, Information Retrieval, Text Generation**

I. Introduction

Large language models (LLMs) have shown strong performance in natural language generation and task processing [1], [2], [3]. Nevertheless, their reliance on static, pre-trained parameters limits their ability to produce accurate responses when the necessary knowledge and up-to-date information are absent [4], [5]. These challenges have prompted growing interest in retrieval-augmented generation (RAG), which integrates information retrieval mechanisms with generative models [6].

Although RAG adheres to a general framework that integrates retrieval and generation (see Figure 1), its implementations differ substantially. Systems may employ diverse retriever and generator architectures, and their combinations are highly flexible. Existing surveys focused on high-level taxonomies or system architectures [7], [8], [9], [10]. However, for both retrieval and generation, formal mathematical definitions offer a clear and rigorous way to describe algorithms. They also provide a foundation for analyzing core algorithmic principles, comparing methods at a fundamental level, and identifying key trade-offs in performance, scalability, and generalization.

This research is supported by the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL).

* Corresponding author: Rui Mao

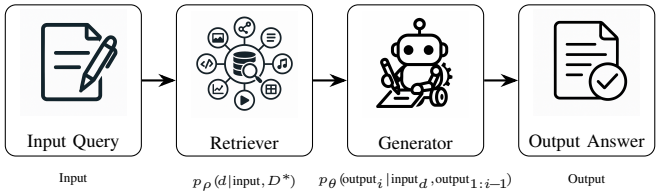

Fig. 1. The general framework of RAG.

This motivates the need for a survey of fundamental algorithms in RAG, which goes beyond surface-level categorization by frameworks or tasks, and instead emphasizes precise formulations, theoretical underpinnings, and algorithmic design patterns. According to the definition of RAG by Lewis et al. [6], we first decompose the framework into two main components: the retriever and generator. An overview of our taxonomy is shown in Figure 2, which summarizes the major categories of retrievers and generators, along with representative methods under each branch. For the retrieval module, we propose a two-level classification. Fundamental retrieval methods operate independently and do not rely on other retrieval strategies. Advanced retrieval methods, in contrast, are built upon fundamental ones and enhance retrievers via additional mechanisms, such as hybridization, iteration, or adaptation. Similarly, we categorize generation methods into two groups. Fundamental generators are classified by their core architecture, e.g., encoder-decoder, decoder-only, and diffusion. Advanced generation refers to techniques that build on these architectures to improve reasoning, alignment, or control, e.g., Chain-of-Thought (CoT) prompting [11] or reinforcement learning (RL) [12]. For each category, we select representative algorithms and present them with mathematical definitions or algorithm pseudocodes under the unified RAG framework. This allows us to explain their working mechanism consistently and rigorously. We also examine how modern RAG systems combine retrievers and generators in practice, and analyze the rationale behind these choices based on their respective strengths and weaknesses. Finally, we summarize the main application areas of RAG, e.g., open-domain question answering, knowledge-grounded dialogue, fact-checking, and scientific reasoning. We review the performance of different methods on representative benchmark datasets, helping readers understand where RAG offers the most value and what challenges remain.

The contributions of this survey are summarized as follows: 1) We propose a unified mathematical framework for modeling the retrieval and generation components in RAG.

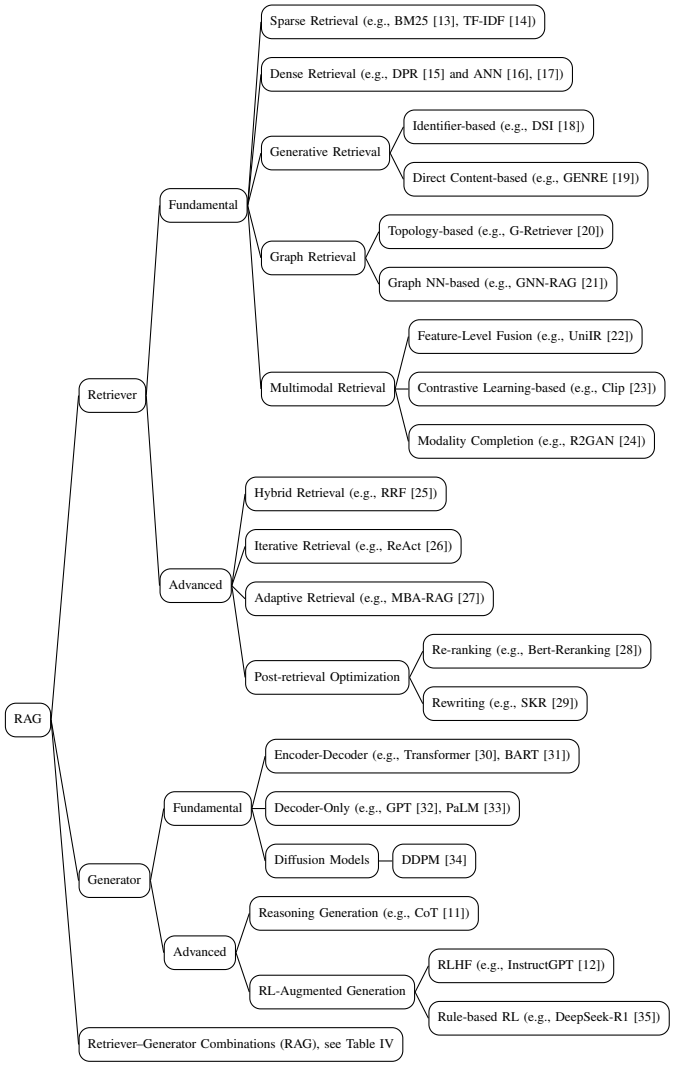

Fig. 2. The taxonomy of RAG in this survey.

This framework supports both with- and without-re-ranking settings, and generalizes common autoregressive generation modes. 2) We present a comprehensive taxonomy of RAG algorithms. Retrieval methods are classified into fundamental and advanced types based on their structural independence, while generation methods are grouped into fundamental architectures and advanced enhancement strategies. Representative algorithms in each category are described with formal equations or pseudocodes under a consistent framework. 3) We analyze how modern RAG systems combine different retrievers and generators in practice, and compare these combinations based on theoretical properties and empirical performance. 4) We review key application domains of RAG, summarize frequently used benchmark datasets, and provide insights into task-specific challenges and performance trends.

The remainder of this paper is organized as follows. Section II outlines the overall RAG formulation and unified equations. Section III reviews retrieval algorithms, covering both fundamental and advanced methods. Section IV focuses on generation methods, including core architectures and advanced generation strategies. Section V examines the practical applications of RAG, the selection of retriever–generator combinations, and performance on representative benchmarks. In each technical section, algorithms are presented in increasing order of complexity, from classical approaches to recent developments. Finally, Section VI concludes the survey.

II. Formal Definition of RAG

A unified mathematical framework that captures the core components of RAG is defined as:

$$p_{\text{RAG}}(\text{output}|\text{input}) \approx \prod_{i=1}^{N} \sum_{d\in D^*} p_\rho(d|\text{input}, D^*)\cdot p_\theta(\text{output}_i|\text{input}_d, \text{output}_{1:i-1}), \tag{1}$$

where d denotes one of the retrieved documents; input denotes the original query; D^*denotes the normalization set used in retrieval, which equals $\mathcal{D}_{\text{top-K}}$ when re-ranking is applied and $\mathcal{C}$ otherwise. $\mathcal{D}_{\text{top-K}}$ denotes the set of top-K retrieved documents and $\mathcal{C}$ denotes knowledge base; output_i is the i-th token of the output sequence; input_d denotes the concatenation of the query and document d; $\text{output}_{1:i-1}$ represents the generated tokens in the former steps. It integrates Retrieval ($p_\rho(\cdot)$) and Generation ($p_\theta(\cdot)$) components, each playing a critical role in incorporating external knowledge into the generation process.

a) Retrieval: The retrieval module aims to identify the most relevant documents $\mathcal{D}_{\text{top-K}}$ from a knowledge base $\mathcal{C}$ given an input. The retriever (parameterized by η) estimates the relevance distribution for each document ($p_\eta(d|\text{input})$). The top-K most relevant documents are retrieved by:

$$\mathcal{D}_{\text{top-K}} = \arg\,\text{top-K}_{d\in\mathcal{C}}\, p_\eta(d|\text{input}). \tag{2}$$

RAG methods often involve a re-ranking step for post-retrieval optimization to refine the retrieved documents before integrating them into the response generation. This operation is consequently reflected in the overall equation. The re-ranking refines the initial ranking of documents in $\mathcal{D}_{\text{top-K}}$ by assigning new scores via a scoring function $f_\rho(\text{input}, d)$. The normalized relevance distribution is given by:

$$p_\rho(d \mid \text{input}, D^*) = \frac{\exp\big((1-\delta)\log p_\eta(d\,|\,\text{input}) + \delta f_\rho(\text{input},d)\big)}{\sum_{d'\in D^*}\exp\big((1-\delta)\log p_\eta(d'\,|\,\text{input}) + \delta f_\rho(\text{input},d')\big)}. \tag{3}$$

where $\delta \in \{0,1\}$ indicates whether re-ranking is applied. $D^* = \mathcal{C}$ when $\delta = 0$; $D^* = \mathcal{D}_{\text{top-K}}$ when $\delta = 1$. When $\delta = 0$, the distribution simplifies to $p_\rho(d \mid \text{input}, \mathcal{C}) = p_\eta(d \mid \text{input})$; when $\delta = 1$, it reduces to the standard re-ranking normalization over $\mathcal{D}_{\text{top-K}}$.

b) Generation: The generation module produces the target output sequence based on the enriched query input_d, which is obtained by combining the original input with the retrieved document d. The overall sequence probability is given by:

$$p_\theta(\text{output}|\text{input}_d) = \prod_{i=1}^{N} p_\theta(\text{output}_i|\text{input}_d, \text{output}_{1:i-1}). \tag{4}$$

The above RAG definition adopts the traditional token-by-token approach for text generation, whereas recent advancements in multimodal research have led to the rise of

alternative generation models, e.g., diffusion models [34]. Diffusion models, originally developed for image and audio generation, iteratively refine noisy inputs into structured outputs via a denoising process, fundamentally differing from autoregressive text generation. Due to their distinct nature, diffusion-based generation models do not fit within the above formulations. We will explain them in Section IV-A3.

III. Retriever Algorithms

In the RAG framework, the retriever selects relevant information from an external knowledge base or corpus to support text generation. The goal is to rank documents based on the input query and provide useful context to improve the accuracy, richness, and reliability of generated content. Here, we classify retrieval approaches into two main categories: Fundamental and Advanced Retrieval Methods. Fundamental Retrieval Methods are independent approaches that do not rely on other retrieval techniques. They form the core of a retrieval system and can retrieve information on their own. It includes sparse retrieval, dense retrieval, generative retrieval, graph retrieval, and multimodal retrieval. In contrast, Advanced Retrieval Methods build on fundamental retrieval methods by applying additional strategies, including hybrid retrieval, iterative retrieval, adaptive retrieval, and some post-retrieval optimization methods, to improve recall or accuracy without changing the fundamental retrieval process.

A. Fundamental Retrieval Methods

1) Sparse Retrieval: Sparse retrieval is a method that represents text as high-dimensional sparse vectors and ranks documents by keyword matching. It mainly uses term frequency-based statistical models to measure the relevance between a query and documents. Common sparse retrieval methods include TF-IDF [36] and BM25 [13].

a) TF-IDF: Term Frequency-Inverse Document Frequency is a widely used sparse retrieval method that evaluates the relevance between a query and a document based on statistical term importance. The TF-IDF score is computed as the product of term frequency (TF) and inverse document frequency (IDF) [14]:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t) \tag{5}$$

where t represents a term; d represents a document. TF measures how frequently a term appears in a document ($\text{TF}(t,d) = \frac{f(t,d)}{|d|}$, where $f(t,d)$ is the raw frequency of term t in document d; $|d|$ represents the total number of words in d). IDF reduces the weight of frequently occurring terms across documents ($\text{IDF}(t) = \log \frac{N}{n_t}$, where N is the total number of documents; n_t is the number of documents containing term t). The TF-IDF score of a given input and a document is

$$\text{score}(\text{input}, d) = \sum_{t \in \text{input}} \text{TF-IDF}(t,d). \tag{6}$$

Next, a Softmax transformation is applied to normalize scores across the retrieved documents:

$$p_\eta(d \mid \text{input}) = \frac{\exp(\text{score}(\text{input}, d))}{\sum_{d' \in \mathcal{D}_{\text{top-K}}} \exp(\text{score}(\text{input}, d'))}. \tag{7}$$

b) BM25: Best Matching 25 is an extension of TF-IDF that introduces non-linear term frequency scaling and document length normalization to improve retrieval performance. It is a ranking function based on the probabilistic relevance framework, designed to rank documents according to their relevance to a given query [13]. Unlike TF-IDF, which assumes a linear relationship between term frequency and relevance, BM25 applies a saturation function to control the influence of frequently occurring terms. The BM25 score for a document d concerning a query input is computed as:

$$\text{BM25}(d,\text{input}) = \sum_{t \in \text{input}} \text{IDF}(t) \cdot \frac{f(t,d) \cdot (k_1+1)}{f(t,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{\text{avgdl}})}, \tag{8}$$

where avgdl is the average document length in the corpus; k_1 is a hyperparameter controlling term frequency saturation; b is a hyperparameter controlling document length normalization.

To integrate BM25 into RAG, the BM25 score is converted into a probability distribution by

$$p_\eta(d \mid \text{input}) = \frac{\exp(\text{BM25}(d, \text{input}))}{\sum_{d' \in \mathcal{D}_{\text{top-K}}} \exp(\text{BM25}(d', \text{input}))}. \tag{9}$$

2) Dense Retrieval: Dense Retrieval, e.g., Dense Passage Retrieval (DPR) [15] typically employs a Dual Encoder architecture to encode queries and documents separately into the same vector space, enabling matching by vector similarity. Compared to Sparse Retrieval, Dense Retrieval relies on high-dimensional dense vectors generated by neural networks, which better capture semantic information [37], [38].

In the dual encoder structure, a query input and document d are mapped to the same vector space via independent encoders: $q = f_{\eta_1}(input) \in \mathbb{R}^{d'}, r_d = g_{\eta_2}(d) \in \mathbb{R}^{d'}$, where $f_\eta(\cdot)$ and $g_\eta(\cdot)$ are query and document encoders, typically BERT-like language models [39]; q and r_d are vector representations, usually normalized before similarity computation; η_1 and η_2 are learnable parameters that are typically distinct for queries and documents. The relevance score is given by: $s(q, r_d) = q^\top r_d$. Common alternatives include cosine similarity or Euclidean distance, whereas dot-product is computationally efficient. The normalized relevance distribution over documents is given by:

$$p_\eta(d|input) = \frac{\exp(s(q, r_d))}{\sum_{d' \in C} \exp(s(q, r_{d'}))} \tag{10}$$

Dense Retrievers are frequently trained by contrastive learning:

$$\mathcal{L} = -\log \frac{\exp(s(q, r_{d^+}))}{\exp(s(q, r_{d^+})) + \sum_{d^- \in N} \exp(s(q, r_{d^-}))}, \tag{11}$$

where d^+ denotes the positive (relevant) document corresponding to the query, and d^- denotes a negative (irrelevant) document sampled from the set N of negative examples.

To enable efficient top-K retrieval over large corpora in vector space, approximate nearest neighbor (ANN) methods partition the vector space. Key categories include hash-based, and vector quantization-based ANN methods.

Locality Sensitive Hashing (LSH) maps similar vectors to the same hash buckets [16], thereby reducing the search space during retrieval. Instead of comparing a query against all vectors in the corpus, LSH allows the system to focus only

on those candidates that fall into the same or nearby buckets, which are likely to be semantically similar. This significantly lowers the computational cost, making it possible to perform approximate retrieval in sublinear time.

Product Quantization (PQ) splits high-dimensional vectors into subvectors for efficient quantization [40]; among its variants, IVFPQ is the most widely used, as it combines an inverted file index with PQ to significantly accelerate large-scale approximate nearest neighbor search. Faiss is an open-source toolkit for vector similarity search that incorporates various indexing methods—including IVFPQ (implemented via the IndexIVFPQ class)—to enable scalable, fast, and memory-efficient approximate nearest neighbor search on billion-scale datasets [17].

3) Generative Retrieval: Generative retrieval directly predicts the identifier or key content of the target document via sequence generation models, bypassing the explicit similarity computation in traditional retrieval. Its core paradigms can be categorized into two types: a) Identifier-based Generative Retrieval: Assigns structured identifiers (e.g., hierarchical encoding) to documents, enabling the model to directly locate documents by generating identifiers. A representative method is Differentiable Search Index (DSI) [18]. b) Direct Content-based Generative Retrieval: Directly generates key textual fragments of documents (e.g., titles or entity names). A typical approach is Generative Entity Retrieval (GENRE) [19]. Another related method is GeAR [41], which generates auxiliary localized content from retrieved documents to enhance retrieval interpretability and fine-grained information access.

a) DSI: DSI aims to transform the entire retrieval pipeline into a single, unified generative model. Instead of following the traditional multi-stage "retrieve-then-rank" approach, DSI encodes the entire corpus into the model's parameters during an indexing stage, and then directly answers queries by generating document identifiers in the retrieval stage. In other words, DSI "memorizes" the corpus and, at inference time, uses this internalized knowledge to map a query directly to a relevant document.

In the indexing stage, DSI learns a generative model $h_\theta(\cdot)$, mapping the textual representation of each document d to a document identifier y (which can be a token sequence):

$$y = h_\theta(d) = [y_1, y_2, \ldots, y_T], \quad y \in \mathcal{Y}, \tag{12}$$

where $h_\theta(\cdot)$ is a sequence-to-sequence model based on Transformer (e.g., T5), with parameters θ [42]; $\mathcal{Y}$ represents the set of all valid document identifiers. The indexing objective is optimized with a standard seq2seq cross-entropy loss predicting docids from document tokens.

In the retrieval stage, given an input, the DSI model directly generates the corresponding document identifier $\hat{y}$:

$$\hat{y} = \text{Decoder}(\text{input}; \theta) = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T]. \tag{13}$$

The probability of generating $\hat{y}$ is given by:

$$p_\theta(\hat{y} \mid \text{input}) = \prod_{t=1}^{T} p_\theta(\hat{y}_t \mid \text{input}, \hat{y}_{<t}). \tag{14}$$

To obtain a ranked list of documents, DSI uses beam search to generate multiple candidate identifiers, and selects:

$$\mathcal{D}_{\text{top-K}} = \arg \text{top-K}_{y \in \mathcal{Y}} \, p_\theta(y|\text{input}), \tag{15}$$

choosing the top-K most probable document identifiers, which are then mapped to their corresponding documents. Retrieval is optimized with a sequence-to-sequence cross-entropy loss predicting docids from queries, and the retrieval and indexing objectives are combined into a joint training objective for simultaneous learning.

b) GENRE: Generative Entity Retrieval redefines the entity retrieval problem as a sequence generation task rather than a traditional multi-class classification problem. Given an input text, e.g., a context containing entity mentions, the goal is to generate the corresponding entity name y, e.g., a Wikipedia title, where the generation process follows an autoregressive token-by-token decoding approach.

Let $y = [y_1, y_2, \ldots, y_T]$ be the sequence of tokens representing the entity name. The GENRE model, parameterized by θ, learns the probability distribution as follows:

$$p_\theta(y \mid \text{input}) = \prod_{t=1}^{T} p_\theta(y_t \mid \text{input}, y_{<t}), \tag{16}$$

where the input consists of the query input and the previously generated tokens $y_{<t}$. Then, GENRE is trained with a standard sequence-to-sequence objective that maximizes the conditional likelihood of the target entity name given the input, following a teacher forcing strategy. During the inference stage, given the input, the model autoregressively generates the entity name like the DSI generation process (Like equation (13)).

To ensure that the generated output corresponds to a valid entity name from the knowledge base, GENRE employs constrained beam search, which restricts decoding to valid prefixes represented by a prefix tree over the candidate set. At inference time, the model explores the search space by maintaining the top-K most likely partial sequences at each decoding step, and returns the top-K completed entity names according to their generation probabilities, this step is also like the equation (15).

In summary, GENRE transforms the entity retrieval problem into an autoregressive generation task, directly generating entity names using a standard seq2seq model with constrained decoding. This approach not only captures fine-grained interactions between the input query and entity names, but also significantly reduces storage requirements and achieves high-quality retrieval results via precise probability calculations.

4) Graph Retrieval: Graph retrieval aims to select the most relevant nodes, subgraphs, or documents from a graph-structured knowledge base for a given query [43]. The goal is to assign relevance scores to candidates and retrieve the most relevant information from the graph. Graph retrieval methods can be classified into two main categories. a) Topology-based retrieval methods: These methods use the graph's connectivity to extract relevant information. A representative method is G-Retriever [20], which applies graph topology and optimization techniques to extract a connected subgraph relevant to the query. b) Graph-deep-learning-based retrieval methods: These methods use graph representation learning, typically with

Graph Neural Networks (GNN), to encode nodes and their neighborhoods into dense embeddings. Relevance is computed by comparing the query embedding with node embeddings. A representative method is GNN-RAG [21].

a) G-Retriever: This method uses a topology-based approach to extract a connected subgraph most relevant to the query. It formulates the retrieval problem as a Prize-Collecting Steiner Tree (PCST) optimization. First, in the prize assignment stage, a candidate node set V_k is obtained using k-nearest neighbors based on text embeddings. Each node d in this set is assigned a prize:

$$\text{prize}(d) = \begin{cases} k - i, & \text{if } d \text{ is ranked } i \text{ in } V_k; \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

where V_k is the set of candidate nodes found via similarity search between the query (input) and graph nodes; i is the ranking position of d in V_k. Each edge e may also be assigned a prize(e) in a similar way based on its relevance to the query. In certain cases, edges can be transformed into virtual nodes to fit the PCST optimization framework. Next, during the PCST Optimization stage, after assigning prizes, G-Retriever solves the PCST optimization problem to extract a connected subgraph $S^* = (V^*, E^*)$ from the graph G. The optimization objective is defined as:

$$S^* = \text{argmax}_{S \subseteq G,\, S \text{ connected}} \big(\sum_{d \in V_S} \text{prize}(d) + \sum_{e \in E_S} \text{prize}(e) - \text{cost}(S) \big), \tag{18}$$

where the edge cost is defined by $\text{cost}(S) = |E_S| \times C_e$. V_S and E_S are the node and edge sets of the selected subgraph S; C_e is a predefined edge cost.

In subgraph retrieval, the extracted subgraph S^* represents the most relevant structure in the graph. The nodes in S^* serve as candidate retrieval results. The retrieval process can be described probabilistically:

$$p_\eta(d|\text{input}) \propto \begin{cases} \exp\big(s_{\text{topo}}(d, \text{input})\big), & \text{if } d \in S^*; \\ 0, & \text{otherwise,} \end{cases} \tag{19}$$

where $s_{\text{topo}}(d, \text{input})$ is the topology-based relevance score between the query input and node d. This equation states that only nodes in the optimal subgraph S^* receive non-zero retrieval probability.

b) GNN-RAG: Given a query input, GNN-RAG first constructs a dense subgraph $\mathcal{G}_q$. Each node d is initialized as

$$h_d^{(0)} = x_d,$$

where x_d denotes the feature vector (e.g., text embedding) of node d. The node representation is then updated through L layers of GNN message passing:

$$h_d^{(l)} = \psi\left(h_d^{(l-1)}, \sum_{j \in N(d)} \omega(\text{input}, r_{d,j})\, m_{dj}^{(l)} \right), \; l = 1, \ldots, L, \tag{20}$$

where $N(d)$ denotes the set of neighbors of node d; $r_{d,j}$ is the relation type on the edge from d to j; $\omega(\text{input}, r)$ measures the relevance between the query and relation; $m_{dj}^{(l)}$ is the message passed from neighbor j at layer l; and $\psi(\cdot, \cdot)$ is a learnable aggregation function. After L layers, we obtain the final representation $h_d \equiv h_d^{(L)}$. The model treats node classification as a softmax over all candidate nodes to compute the retrieval probability:

$$p_\eta(d \mid \text{input}) = \frac{\exp\left(\mathbf{w}^\top h_d\right)}{\sum_{d' \in \mathcal{C}} \exp\left(\mathbf{w}^\top h_{d'}\right)}, \tag{21}$$

where $\mathbf{w}$ is a trainable classification weight vector and $\mathcal{C}$ denotes the candidate node set. The nodes with the highest $p_\eta(d \mid \text{input})$ are returned as retrieval results, along with their shortest paths to the query entity for downstream reasoning and generation.

In addition to these two methods, some methods combine both strategies. For example, hybrid approaches may apply k-hop subgraph extraction (topology-based retrieval) and then refine the candidates using GNN-based encoding and ranking [44]. This combination benefits from both the graph's structural properties and the semantic richness.

5) Multimodal Retrieval: Multimodal retrieval aims to process and understand data from multiple modalities, such as text, images, audio, and video, to meet diverse user needs. By integrating and aligning different modalities, the system enables cross-modal retrieval and matching within a unified framework. Multimodal retrieval methods can be categorized into three types. a) Feature-Level Fusion Retrieval: This approach integrates features from different modalities during the feature extraction process. For instance, in the UniIR model, image and text features are merged within a unified framework and jointly optimized, resulting in a comprehensive multimodal representation that leverages complementary information to enhance retrieval performance [22]. b) Contrastive Learning-based Retrieval: This approach utilizes separate encoders for each modality and aligns their representations in a shared embedding space via contrastive objectives. For example, the CLIP model [45] trains independent image and text encoders on large-scale datasets using a contrastive loss, effectively bridging the semantic gap between modalities. This enables robust cross-modal retrieval [23], such as performing image search with textual queries, without directly fusing raw features during extraction. c) Modality Completion Retrieval: This method uses generative models to bridge gaps between modalities or to supplement missing modality information. An example is the R2GAN model [24], which employs a generative adversarial network (GAN) [46] to create dish images from recipe text. The generated images, together with real images, are used to train a cross-modal embedding space, improving retrieval performance for food-related queries. This approach not only enhances cross-modal feature representation but also provides an intuitive way to explain retrieval results through generated content.

a) UniIR: UniIR is a unified framework designed to handle various retrieval tasks across different modalities, such as text and images. It interprets user instructions to perform specific retrieval tasks, aiming to retrieve the most relevant contents $\mathcal{D}_{\text{top-K}}$ from a knowledge base $\mathcal{C}$ based on a given input. UniIR employs two main fusion strategies to compute

the relevance score $p_\eta(d \mid \text{input})$, namely Score-Level Fusion and Feature-Level Fusion.

In the Score-Level Fusion approach, separated encoders process different modalities. Their outputs are combined at the score level. For example, for a query consisting of an image input_i and text input_t with an instruction $\text{input}_{\text{inst}}$, the combined query representation is given by:

$$f(\text{input}_i, \text{input}_t, \text{input}_{\text{inst}}) = w_1\, f_I(\text{input}_i) + w_2\, f_T(\text{input}_t, \text{input}_{\text{inst}}) \tag{22}$$

where $f_I(\cdot)$ and $f_T(\cdot)$ are image and text encoders, respectively; w denotes learnable weights. For a candidate document with image d_i and text d_t, its representation is given by:

$$f(d_i, d_t) = w_3\, f_I(d_i) + w_4\, f_T(d_t), \tag{23}$$

Then, the relevance score is computed as the dot product of the query and candidate representations:

$$s_{\text{input},d} = f(\text{input}_i, \text{input}_t, \text{input}_{\text{inst}})^\top \cdot f(d_i, d_t). \tag{24}$$

On the other hand, the Feature-Level Fusion method integrates features from different modalities before computing the relevance score. The query representation is generated by a multimodal encoder $f_{\text{MIX}}(\cdot)$ which processes the combined input, $f_{\text{MIX}}(\text{input}_i, \text{input}_t, \text{input}_{\text{inst}})$. Similarly, the candidate document representation is also given by a multimodal encoder, $f_{\text{MIX}}(d_i, d_t)$. Then, the relevance score is the dot product of the multimodal representations of a query and a document:

$$s_{\text{input},d} = f_{\text{MIX}}(\text{input}_i, \text{input}_t, \text{input}_{\text{inst}})^\top \cdot f_{\text{MIX}}(d_i, d_t). \tag{25}$$

During the training process, UniIR is trained using a contrastive loss that maximizes the relevance scores of correct query–candidate pairs while minimizing those of incorrect pairs. This training enables the model to effectively handle diverse retrieval tasks across multiple modalities. Thus, the probability $p_\eta(d \mid \text{input})$ can be approximated by normalizing the similarity scores over all candidate contents in $\mathcal{C}$:

$$p_\eta(d \mid \text{input}) = \frac{\exp\big(s_{\text{input},d}\big)}{\sum_{d' \in \mathcal{C}} \exp\big(s_{\text{input},d'}\big)}, \tag{26}$$

where $s_{\text{input},d}$ is the similarity score between the query and candidate contents d. The top-K most relevant contents $\mathcal{D}_{\text{top-K}}$ are then selected based on these probabilities.

b) CLIP: In CLIP, contrastive learning is employed to simultaneously train an image encoder $f_I(\cdot)$ and a text encoder $f_T(\cdot)$ so that the representations of paired images and texts are brought close together in a shared embedding space, while those of non-matching pairs are pushed apart. This training mechanism enables the model to learn cross-modal representations that can be directly used for retrieval tasks such as retrieving images based on text queries (or vice versa).

During the feature extraction and projection phase, for a given image I and text T, features are first extracted using their respective encoders:

$$I_f = f_I(I), \quad T_f = f_T(T) \tag{27}$$

Then, linear mappings (W_I and W_T) are applied to project these features into a common multimodal embedding space, followed by L_2 normalization:

$$I_e = \frac{W_I I_f}{\|W_I I_f\|}, \quad T_e = \frac{W_T T_f}{\|W_T T_f\|}. \tag{28}$$

To measure the relevance between the image and text, the normalized cosine similarity ($\langle \cdot, \cdot \rangle$) is computed and scaled by a learnable temperature parameter τ: $s(I, T) = \exp(\tau) \cdot \langle I_e, T_e \rangle$. During the pre-training phase, for a batch of N image-text pairs, a similarity matrix S of size $N \times N$ is constructed, where the (i, j)-th element is given by: $S_{ij} = \exp(\tau) \cdot \langle I_{e,i}, T_{e,j} \rangle$.

To distinguish between the true pairs and false pairs, a symmetric cross-entropy loss is applied in both directions. For the image-to-text direction, the loss is defined as:

$$\mathcal{L}_{I \to T} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\tau) \cdot \langle I_{e,i}, T_{e,i} \rangle}{\sum_{j=1}^{N} \exp(\tau) \cdot \langle I_{e,i}, T_{e,j} \rangle}. \tag{29}$$

For the text-to-image direction, the loss is given by:

$$\mathcal{L}_{T \to I} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\exp(\tau) \cdot \langle T_{e,i}, I_{e,i} \rangle}{\sum_{j=1}^{N} \exp(\tau) \cdot \langle T_{e,i}, I_{e,j} \rangle}. \tag{30}$$

The overall los $\mathcal{L}$ is the average of these two directional losses. After pre-training, the learned cross-modal embeddings can be directly applied to retrieval tasks. Taking a query input (e.g. text or image) as an example, its embedding is obtained via a query encoder $f_q(\cdot)$ with projection and normalization:

$$q = \frac{W_q\, f_q(\text{input})}{\|W_q\, f_q(\text{input})\|}. \tag{31}$$

For each candidate $d \in \mathcal{C}$, e.g., an image or text, its embedding is obtained by a document encoder ($f_d(\cdot)$):

$$r_d = \frac{W_d f_d(d)}{\|W_d f_d(d)\|}. \tag{32}$$

The relevance score between the query and each candidate is computed using the scaled cosine similarity:

$$\text{score}(d|\text{input}) = \exp(\tau) \cdot \langle q, r_d \rangle. \tag{33}$$

By applying a softmax function over the scores of all candidates, a probability distribution is given by:

$$p_\eta(d|\text{input}) = \frac{\exp(\tau) \cdot \langle q, r_d \rangle}{\sum_{d' \in \mathcal{C}} \exp(\tau) \cdot \langle q, r_{d'} \rangle}. \tag{34}$$

The top-K most relevant documents are then could be selected.

CLIP jointly trains an image encoder and a text encoder to learn a shared multimodal embedding space. It computes relevance via normalized cosine similarity with temperature scaling, and selects top-K candidates by applying softmax over similarity scores, enabling efficient semantic cross-modal retrieval.

c) R2GAN: The overall objective of R2GAN is to learn a common embedding space for recipes and images, while also generating images from recipe embeddings for interpretable retrieval. In the Feature Extraction and Embedding Mapping stage, for a recipe T and an image I, the raw features are first extracted using their respective encoders same as the CLIP methods and also project these features into a common space as equation (27) and equation (28) show. Then we could also get I_e and T_e. Here $f_T(\cdot)$ employs a bidirectional LSTM and hierarchical LSTM to encode the recipe text, $f_I(\cdot)$ is a modified ResNet-50 for image feature extraction. In the Cross-Modal Image Reconstruction stage, a generator G is used to reconstruct images from either modality, which both enhances the compatibility of the learned embeddings and provides visual explanations for retrieval:

$$v_f^T = G(T_e), \quad v_f^I = G(I_e), \tag{35}$$

where v_f^T is the image generated from a recipe embedding; v_f^I is the image reconstructed from an image embedding. To ensure that both the embedding space and the reconstructed image space can discriminate between positive and negative samples, a two-level ranking loss is introduced. Let q, p_e, and n_e denote the query, positive, and negative embeddings, respectively, and let input$_v$, p_v, and n_v be the corresponding reconstructed images. The loss is defined as:

$$\begin{aligned} \mathcal{L}_{\text{rank}} &= \max\{\langle q, n_e\rangle - \langle q, p_e\rangle + \alpha_1,\, 0\} \\ &+ \mu\, \max\{d(\text{input}_v, p_v) - d(\text{input}_v, n_v) + \alpha_2,\, 0\}, \end{aligned} \tag{36}$$

where $\langle a, b\rangle$ denotes cosine similarity in the embedding space, $d(a, b)$ denotes Euclidean distance in the image space, α_1 and α_2 are margin hyperparameters, and μ is a trade-off factor.

R2GAN introduces two discriminators, D_1 and D_2. The first loss term, $\mathcal{L}_{D_1}$, measures the ability of D_1 to distinguish between real images and those reconstructed from image embeddings. Similarly, $\mathcal{L}_{D_2}$ quantifies how well D_2 can differentiate between images generated from image embeddings and those generated from recipe embeddings. For the generator, the adversarial objective $\mathcal{L}_G$ combines feedback from both discriminators, encouraging the generator to simultaneously fool D_1 and D_2. To ensure that the reconstructed images preserve as much information from the originals as possible, the reconstruction loss $\mathcal{L}_{\text{recon}}$ is introduced, which penalizes discrepancies both in the feature space and the image space. In addition, to incorporate high-level semantic information, the semantic loss $\mathcal{L}_{\text{sem}}$ is employed using a cross-entropy formulation, aiming to align the generated representations with the correct food category labels. Then, the overall loss for the embedding and semantic modules is:$\mathcal{L}_{\text{full}} = \mathcal{L}_{\text{rank}} + \gamma\,\mathcal{L}_{\text{recon}} + \lambda\,\mathcal{L}_{\text{sem}}$. The full loss for updating the generator is defined as: $\mathcal{L}_{G_{\text{full}}} = \mathcal{L}_G + \delta\,\mathcal{L}_{\text{recon}}$ where γ, λ, and δ are balancing hyperparameters.

During retrieval, for a given query input (e.g., a recipe or an image), its embedding is same as equation (31). where $f_q(\cdot)$ here also denotes the corresponding encoder (f_T for recipes or f_I for images). For each candidate document $d \in \mathcal{C}$ (recipe or image), the embedding is same as equation (32). The relevance score between the query and a candidate is computed as a scaled cosine similarity, and the resulting probability distribution is also same as equation (34). Then top-K candidates cloud be selected. Furthermore, for recipe queries, the reconstructed image can be used to provide an interpretable explanation for the retrieval results.

R2GAN exemplifies the concept of Modality Completion Retrieval with GAN to generate dish images from recipe texts. These generated images, used alongside real images, facilitate the learning of a robust cross-modal embedding space that not only improves retrieval performance for food-related queries but also provides intuitive visual explanations for the results.

Algorithm 1 Hybrid Retrieval with Complementarity Objectives and Candidate Selection

Require: Query q, Document Collection $\mathcal{D}$, Lexical Retriever S, Semantic Retriever D, Orthogonality Module O, balance factor λ, top-K parameter K
Ensure: top-K ranked documents for query q
for each document $d \in \mathcal{D}$ **do**
  $s \leftarrow S(q, d)$ ▷ Compute lexical similarity (e.g., BM25)
  $r \leftarrow D(q, d)$ ▷ Compute semantic similarity (e.g., inner product of BERT embeddings)
  $Score_{dual}(q, d) \leftarrow s + \lambda \cdot r$
end for
$L_{ortho} \leftarrow O(\{s, r\}_{d\in\mathcal{D}})$ ▷ Compute orthogonality loss to reduce redundant features
Update model parameters by minimizing $L_{total} = L_{rel} + L_{ortho}$
return top-K documents from $\mathcal{D}$ ranked by $Score_{dual}(q, d)$

B. Advanced Retrieval Methods

1) Hybrid Retrieval: Hybrid retrieval methods aim to combine the strengths of different basic retrieval methods to improve both recall and accuracy. A representative method (Algorithm 1) in this area combines a sparse retriever (such as BM25) with a dense retriever (such as DPR) [47]. In this work, dense retrieval methods encode queries and documents into dense vectors capturing semantic meaning, whereas sparse retrieval methods leverage lexical matches. The proposed hybrid approach combines these two methods, enhanced by embedding-level and input-level orthogonality constraints to maximize complementarity between sparse and dense retrieval.

2) Iterative Retrieval: Iterative retrieval methods allow the retrieval process to interact with the reasoning process. A representative method (Algorithm 2) is based on the ReAct framework [26], where the model alternates between reasoning (generating a CoT) and acting (generating a new query) based on previous retrieval results. This process continues until sufficient information is gathered.

3) Adaptive Retrieval: Adaptive retrieval dynamically adjusts the retrieval strategy based on the query or task requirements. A representative method (Algorithm 3) is MBA-RAG [27], using a multi-armed bandit model to select the best

Algorithm 2 Iterative Retrieval using ReAct

Require: Query q, maximum iterations N, language model M, retriever R
Ensure: Final answer based on retrieval and reasoning
Initialize $context \leftarrow []$
for $i = 1$ to N **do**
  $thought \leftarrow M$.reason$(q, context)$ ▷ Generate a chain-of-thought
  if M.decide_stop$(thought)$ **then**
    break
  end if
  $new\_query \leftarrow M$.generate_query$(thought)$ ▷ Generate a new sub-query
  $results \leftarrow R.retrieve(new\_query)$
  Append $results$ to $context$
end for
Use the final $context$ and chain-of-thought to generate the answer

Algorithm 3 Adaptive Retrieval with Multi-Armed Bandit

Require: Query q, a set of retrieval modules $\{M_1, M_2, \ldots, M_k\}$, policy selector $Policy$
Ensure: Retrieval result R
$i \leftarrow Policy.select(q)$ ▷ Choose a module based on query features
$R \leftarrow M_i.retrieve(q)$
Obtain feedback (e.g., accuracy, cost)
Update $Policy$ using the feedback
return R

Algorithm 4 Supportiveness-based Knowledge Rewriting (SKR)

Require: Query q, retrieved documents $D = \{d_1, d_2, \ldots, d_n\}$, rewriting model T, supportiveness evaluator E, threshold τ
Ensure: Rewritten knowledge set D'
$D' \leftarrow \emptyset$
for each document $d \in D$ **do**
  $r \leftarrow T$.rewrite(d, q) ▷ Generate a rewritten version of d using the query q
  $s \leftarrow E$.score(q, r) ▷ Compute the supportiveness score of r for query q
  if $s \geq \tau$ **then**
    $D' \leftarrow D' \cup \{r\}$
  else
    Discard r
  end if
end for
return D'

retrieval module (e.g., BM25, DPR, or multi-hop retrieval) for a given query. The model learns from feedback to balance between different strategies, improving efficiency and accuracy.

4) Post-retrieval Optimization: Post-retrieval augmentation enhances retrieved result quality and includes two subtypes: Rewriting and Re-ranking.

Rewriting methods refine retrieved documents to better align them with the needs of the generation model. For example, SKR (Supportiveness-based Knowledge Rewriting in Algorithm 4) [29] condenses each document by filtering out noise and retaining only the content that most effectively supports the query. This yields more concise and focused documents for downstream generation. Other rewriting approaches not only include extractive filtering and multi-document summarization but also extend to prompt compression techniques, since the retrieved documents are used as prompts for the generation model, compressing them can be seen as a form of rewriting. For instance, the Gist algorithm compresses lengthy prompts into a compact set of "gist" tokens that encapsulate the core information with minimal redundancy [48]. AutoCompressors transform extended context documents into concise summary vectors that serve as soft prompts, ensuring that only the most relevant content is passed to the generation model [49].

Re-ranking methods use a more detailed matching model to re-order the initially retrieved documents. A very classic approach is PageRank, which assigns a numerical value to each web page based on the quantity and quality of links pointing to it, thereby measuring its relative importance on the web [50]. Recently, a widely used re-ranking method is called the BERT-based Passage Re-ranking model, where a cross-encoder model (e.g., BERT) computes a relevance score for each candidate document by concatenating the query with the document and then reorders them based on these scores [28]. BERT-based Passage Re-ranking employs a Cross-Encoder architecture where a query and a candidate passage are jointly encoded by a BERT model. Given a query (denoted as input) and a candidate passage d, the model first constructs a single input sequence by concatenating them with special tokens, e.g., "[CLS] input [SEP] d [SEP]". Due to length constraints, the query is truncated to at most 64 tokens and the passage is truncated so that the overall sequence length does not exceed 512 tokens. The concatenated sequence is then fed into BERT to produce contextualized representations. The output ($\mathbf{h}_{\text{[CLS]}}$), corresponding to the [CLS] token, is used to compute the relevance score via a classification layer:

$$f_\rho(\text{input}, d) = \sigma\Big(\mathbf{W} \cdot \mathbf{h}_{\text{[CLS]}} + b\Big), \tag{37}$$

where $\mathbf{W} \in \mathbb{R}^{1\times H}$ and $b \in \mathbb{R}$ are the trainable parameters (collectively denoted by ρ) of the classification layer; $\sigma(\cdot)$ is the sigmoid activation function; H is the hidden dimension of BERT; $f_\rho(\text{input}, d) \in [0, 1]$ represents the probability that passage d is relevant to the query. For a set of candidate passages C, the re-ranking is performed by computing the relevance score for each passage:

$$s_d = f_\rho(\text{input}, d), \quad \forall\, d \in C, \tag{38}$$

and sorting the passages in descending order of s_d to select the top-N passages. The model is trained using the binary cross-entropy loss defined over the candidate set:

$$\mathcal{L} = -\sum_{d\in C} \Big[y_d \log f_\rho(\text{input}, d) + (1-y_d) \log\big(1 - f_\rho(\text{input}, d)\big)\Big], \tag{39}$$

where $y_d = 1$ if d is relevant to the query, otherwise $y_d = 0$.

TABLE I
THE COMPARISON OF DIFFERENT RETRIEVAL METHODS.

	Method	Advantages and Disadvantages
Fundamental	Sparse Retrieval	**Pro:** Simple and fast with low computational overhead. Interpretable results. No training required. **Cons:** Weak in capturing semantics. Sensitive to lexical variations. Ineffective for complex queries.
	Dense Retrieval	**Pros:** Strong semantic understanding. Good generalization across tasks. Supports end-to-end training. **Cons:** High training and inference cost. Low interpretability. May require domain-specific fine-tuning.
	Generative Retrieval	**Pros:** Explicit index is not required. Flexible and end-to-end. Integrates well with generation models. **Cons:** Hard to interpret and trace. Susceptible to generation errors. Less scalable for large corpora.
	Graph Retrieval	**Pros:** Capture structured relationships. Multi-hop reasoning. Integrate knowledge graphs effectively. **Cons:** High graph construction and inference cost. Complex pipeline setup. Sensitive to graph quality.
	Multimodal Retrieval	**Pros:** Support cross-modal search. Aligns different modalities in shared space. Multimedia capability. **Cons:** Large-scale multimodal data dependency. Complex alignment and fusion of modalities.
Advanced	Hybrid Retrieval	**Pros:** Combine strengths of different retrieval paradigms. More robust across query types. Flexible integration for diverse tasks. **Cons:** Fusion strategies are hard to tune. Increased computational cost. Higher system complexity.
	Iterative Retrieval	**Pros:** Multi-step reasoning. Refine results step-by-step. Dynamic interaction. **Cons:** Slower inference due to iterations. Susceptible to reasoning drift. Error propagation.
	Adaptive Retrieval	**Pros:** Dynamic retrieval strategy. Resource-efficient in multitask scenarios. Heterogeneous inputs. **Cons:** Sophisticated policy learning. High implementation complexity. Unseen task generalization.
	Re-Ranking	**Pros:** Remarkable precision enhancement. Exploit deep contextual representations. **Cons:** High computational cost. Dependent on initial retrieval quality. Slower response time.
	Re-Rewriting	**Pros:** Refine input for downstream generation. Long-context compression. Irrelevant or redundant text removal. **Cons:** Key information missing. Error propagation.

C. Summary

Table I provides a structured summary of current retrieval strategies, categorized into fundamental methods and advanced techniques. Among the fundamental methods, sparse retrieval remains useful for traditional information retrieval tasks due to its simplicity, efficiency, and lack of training requirements [51]. However, its reliance on exact lexical matching makes it inadequate for capturing deep semantic meaning, limiting its effectiveness for complex natural language queries. In contrast, dense retrieval leverages learned semantic embeddings to better model user intent, making it well-suited for tasks like open-domain question answering [52], albeit with significantly higher training and inference costs. Generative retrieval eliminates the need for explicit indexing by retrieving directly through generation, offering end-to-end flexibility, particularly in entity linking and small-scale corpora [53]. Yet, its lack of interpretability and susceptibility to generation errors pose notable challenges. Graph-based retrieval models entity relationships via knowledge graphs, enabling multi-hop reasoning and structured inference [54], which are especially beneficial in domains like scientific QA and biomedical applications. However, the cost of graph construction and reasoning limits its scalability. Multimodal retrieval, which supports queries across different modalities, is critical for multimedia applications [55], but aligning and training across modalities remains a significant challenge.

Among advanced methods, hybrid retrieval integrates the strengths of multiple paradigms, such as sparse and dense retrieval, resulting in greater robustness across diverse queries. Nevertheless, the fusion strategy can be difficult to tune and optimize. Iterative retrieval enables step-by-step refinement via multi-turn reasoning, improving performance on complex tasks such as multi-hop QA. However, it often incurs longer inference time and may suffer from reasoning drift. Adaptive retrieval dynamically selects the optimal strategy for each query, enhancing resource efficiency in multi-task and resource-constrained settings. However, it requires sophisticated policy learning and may struggle to generalize to unseen queries. Re-ranking serves as a powerful post-retrieval step that significantly improves precision by applying deep semantic models to a small set of retrieved candidates, though it introduces additional computational overhead and latency. Rewriting, on the other hand, preprocesses and restructures long documents to facilitate downstream generation, which is especially valuable for compressing lengthy contexts. However, it carries the risk of information loss and semantic drift during transformation.

Overall, each retrieval method presents unique strengths and limitations. Practical deployment requires careful consideration of application requirements, task complexity, and computational constraints.

IV. GENERATION ALGORITHMS

Within the RAG framework, the generator produces target content conditioned on the input and the retrieved contextual information. Existing generation methods are categorized into two complementary paradigms, namely fundamental generation methods and advanced generation Methods.

Fundamental generation approaches establish core probabilistic frameworks, encompassing three main categories. Encoder-decoder models, such as Transformer [30], T5 [42], and BART [31], combine bidirectional encoding with autoregressive decoding. Decoder-only models, including GPT [12], [32], [56], [57], [58], PaLM [33], LLama [59], [60], [61], Mixtral [62], and DeepSeek [35], [63], [64], [65], generate sequences token-by-token via chain rule decomposition. Lastly, diffusion models, such as DDPM [34] and LDMs [66], reconstruct data through iterative denoising over T steps, typically represented as $\mathbf{x}_{t-1} = f(\mathbf{x}_t, \epsilon_\theta(\mathbf{x}_t, t))$.

Advanced generation methods improve output quality through algorithmic innovations. One prominent direction is reasoning-based generation, exemplified by CoT [11] and True-of-Thought (ToT) [67], which guide models to produce intermediate reasoning steps, thereby emulating human-like sequential problem solving prior to generating a final response. Another line of work leverages RL, either through alignment with human feedback, as in Instruct-GPT [12], or by optimizing predefined reward functions, as in DeepSeek-R1 [35], to progressively enhance model reasoning ability and coherence.

A. Fundamental Generation Methods

1) Encoder-Decoder Models: This is a fundamental paradigm in natural language generation tasks. The core idea is to encode the input sequence and relevant information into a continuous representation using an encoder, followed by a decoder that generates the target sequence step by step.

Early neural network-based sequence generation tasks primarily relied on sequence-to-sequence (Seq2Seq) [68] architectures and recurrent neural networks (RNNs), including Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) [69] models. RNN-based seq2seq models process input sequences recursively and generate output sequences step by step, effectively capturing temporal dependencies. However, due to their inherently sequential nature, they suffer from poor parallelizability and struggle to model long-range dependencies, leading to gradient vanishing issues in long sequences. To mitigate the inefficiencies of RNNs, Convolutional Neural Networks (CNNs) [70] were introduced as an alternative. ConvS2S [71] replaces recurrent units with convolutional layers, enabling parallel processing and improving computational efficiency. By stacking multiple convolutional layers, these models gradually expand their receptive fields to capture longer-range dependencies. While these early models provided a foundation for sequence generation, their limitations in parallelization and long-range dependency modeling motivated further advancements in architecture design. These challenges ultimately led to the development of the Transformer model [30], which fully eliminates recurrence and convolution, achieving both efficient parallel computation and improved dependency modeling. The Transformer model is an attention-based Encoder-Decoder structure that captures long-range dependencies and eliminates the computational constraints of traditional RNNs and CNNs. The elemental computing unit of the Transformer model is the multi-head attention mechanism that can capture long-range dependencies in parallel, allowing for richer contextual representations, and significantly accelerating training compared to the sequential nature of RNNs.

In RAG, the input is first constructed by concatenating the query and the top-K retrieved documents: $\text{input}_d = [\text{query}; \text{Document 1}; \ldots; \text{Document K}]$. The goal of the model is to generate an answer. We obtain the encoder output as: $h_{\text{enc}} = \text{Encoder}(\text{input}_d)$. The generation probability is defined as:

$$p_\theta(\text{output} \mid \text{input}_d) = \prod_{i=1}^{N} p_\theta(\text{output}_i \mid h_{\text{enc}}, \text{output}_{1:i-1}). \tag{40}$$

At each decoding step i, the model predicts the next token output_i conditioned on the encoded input h_{enc} and the previously generated tokens $\text{output}_{1:i-1}$.

The training objective minimizes negative log-likelihood:

$$\mathcal{L} = -\log p_\theta(\text{output} \mid \text{input}_d). \tag{41}$$

This objective is optimized using teacher-forcing, where the ground-truth prefix is provided during training.

In recent years, encoder-decoder generators are commonly based on pre-trained language models. This is because pre-trained language models offer strong generalization capabilities and rich linguistic knowledge, which significantly enhance the quality of generated outputs when conditioned on retrieved context. Widely used encoder-decoder pre-trained language models include T5, and BART. The T5 (Text-to-Text Transfer Transformer) model unifies all tasks into a text-to-text format, enabling cross-task transfer learning. The BART (Bidirectional and Auto-Regressive Transformers) model combines the strengths of bidirectional encoders and autoregressive decoders. Through sequence reconstruction pre-training, BART excels in tasks like text generation and summarization.

2) Decoder-Only Models: Decoder-Only models do not use a separate encoder. Instead, they process the input and output as a single concatenated sequence. At each generation step, the model uses masked self-attention to access only the input_d and previously generated tokens.

The generation probability is formulated as:

$$p_\theta(\text{output} \mid \text{input}_d) = \prod_{i=1}^{N} p_\theta(\text{output}_i \mid \text{input}_d, \text{output}_{1:i-1}). \tag{42}$$

At each step i, the decoder predicts output_i based on input_d and $\text{output}_{1:i-1}$, using a stack of masked self-attention layers.

The training objective is similar to the encoder-decoder case (see Equation (41)), using teacher-forcing by feeding the ground-truth prefix to the decoder. Through stacked decoder layers, it models the conditional probability in an end-to-end manner. As both Encoder-Decoder and Decoder-Only models follow the autoregressive generation paradigm. However, given the power of LLMs, fine-tuning a Decoder-Only LLM for RAG is not an indispensable process. By reformatting diverse tasks, e.g., text classification as generating class labels, question answering as generating answer text, into a unified sequence-to-sequence paradigm, it achieves versatility across NLP applications. The ability of the Decoder-Only architecture to model both input and generated sequences in a unified manner has established it as the mainstream choice for LLMs.

3) Diffusion Models: Diffusion models [72], [66], [73], [74], [75], [76], [77], [78], [79] are a class of generative methods that create data by gradually turning random noise into meaningful outputs. They follow a step-by-step denoising process, learning to recover data from noise through repeated refinement. They use two key processes: a forward process that adds Gaussian noise to the data over many steps, and a reverse process that learns how to remove this noise and restore the original input. Unlike autoregressive models, which produce data one part at a time based on earlier outputs, diffusion models generate the whole sample at once. They operate in continuous spaces and improve the full structure of the data in parallel. This approach makes diffusion models effective at generating high-quality signals such as images and audio. Foundational models such as Denoising Diffusion Probabilistic Models (DDPM) have built the theoretical base for later improvements [34].

DDPMs are a class of generative models based on Markov chains, as shown in Algorithm 5. They first use a set of predefined variance schedules $\{\beta_t\}$ to progressively add Gaussian noise to clean data x_0 in closed form (forward diffusion);

Algorithm 5 DDPM Training and Sampling

Require: Clean data set $\mathcal{X} = \{x_0\}$, diffusion steps T, variance schedule $\{\beta_t\}_{t=1}^{T}$, noise-prediction network ϵ_θ
Ensure: Trained noise predictor ϵ_θ; sampling procedure to generate new data

procedure TRAINDDPM($\mathcal{X}, T, \{\beta_t\}, \epsilon_\theta$)
  Precompute schedules: $\alpha_t \leftarrow 1-\beta_t$, $\bar{\alpha}_t \leftarrow \prod_{s=1}^{t} \alpha_s$
  for each minibatch $x_0 \in \mathcal{X}$ **do**
    Sample timestep $t \sim \text{Uniform}\{1, \ldots, T\}$
    Sample noise $\epsilon \sim \mathcal{N}(0, I)$
    Construct noisy example:

$$x_t \leftarrow \sqrt{\bar{\alpha}_t}\, x_0 \; + \; \sqrt{1-\bar{\alpha}_t}\, \epsilon$$

    Predict noise: $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t,\, t)$
    Compute loss: $L \leftarrow \|\epsilon - \hat{\epsilon}\|^2$
    Update θ via backprop on L
  end for
  return trained ϵ_θ
end procedure
procedure SAMPLEDDPM($\epsilon_\theta, T, \{\beta_t\}$)
  Precompute schedules as above
  Initialize $x_T \sim \mathcal{N}(0, I)$
  for $t = T,\, T-1,\, \ldots,\, 1$ **do**
    Predict noise: $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t,\, t)$
    Sample $z \sim \mathcal{N}(0, I)$ if $t > 1$, else $z \leftarrow 0$
    Denoising update:

$$x_{t-1} \leftarrow \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\epsilon}\Big) + \sqrt{\beta_t}\, z$$

  end for
  return x_0 ▷ Generated sample
end procedure

Algorithm 6 CoT Predict

Require: Model $model$, Query q, examples $examples$
Ensure: Output containing the chain-of-thought and final answer

$prompt \leftarrow$ ""
for all $ex \in examples$ **do**
  $prompt \leftarrow prompt +$ `"Q: "` $+ ex.q +$ `"\nA: "` $+ ex.chain +$ `" The answer is "` $+ ex.answer +$ `"\n\n"`
end for
$prompt \leftarrow prompt +$ `"Q: "` $+ q +$ `"\nA: "`
$output \leftarrow model.generate(prompt)$
return $output$

then, they train a neural network $\epsilon_\theta(x_t, t)$ to predict the noise and minimize the mean squared error between the prediction and the true noise (reverse denoising). In the sampling phase, starting from pure noise $x_T \sim \mathcal{N}(0, I)$, the network iteratively denoises in reverse order, eventually recovering high-quality samples x_0. This "fixed noise injection + learned denoising" approach has demonstrated excellent results in image generation tasks.

The strength of diffusion models lies in their globally coherent generation via iterative refinement, which avoids the autoregressive error accumulation seen in Decoder-Only models. By modeling data distributions via gradual denoising, rather than token-by-token prediction, they achieve superior performance in continuous signal generation (e.g., images, audio) [80], [81], [82], [83], [84] and offer inherent stability during training, circumventing mode collapse issues of GANs.

B. Advanced Generation Methods

1) Reasoning Generation: Reasoning generation refers to the process in which a model produces a sequence of intermediate logical steps prior to arriving at a final answer. The underlying principle is that, for complex tasks, the model should not rely solely on direct input-to-output mapping. Rather, it should emulate human-like step-by-step thinking, progressively narrowing the solution space and enhancing answer accuracy through structured intermediate inferences. CoT [11] is a representative technique in this domain, which effectively avoids the high data cost of pure supervised reasoning methods [85] and the weakness of standard prompting [86] on complex reasoning tasks. The core idea of CoT is to break down the problem gradually during the reasoning process. The model, following the given prompt, first produces a series of intermediate reasoning steps, and then gives the final answer. Algorithm 6 is a pseudocode example that demonstrates how to apply the CoT method in an LLM to solve a problem. The procedure begins by constructing a few-shot prompt that includes multiple examples, each comprising a question, a corresponding reasoning process, and a final answer. A new question is then appended to this prompt, and the model is prompted to generate a complete response. The expected output includes both the intermediate reasoning steps and the final answer. This few-shot prompting strategy represents a standard implementation of the CoT approach.

Although CoT significantly improves reasoning, a single reasoning path may sometimes lead to random biases or errors. Self-consistency [87] is an extension and improvement of CoT. Instead of relying on a single generated reasoning chain, a self-consistency model generates multiple reasoning chains and then compares the answers obtained from these chains. The most consistent answer is chosen as the final result. Algorithm 7 is a pseudocode example that demonstrates the self-consistency reasoning process. Given a model `CoT_model` capable of performing CoT reasoning through few-shot prompting, the method generates multiple candidate outputs via sampling and selects the final answer based on a majority vote among the sampled responses. In Algorithm 7, the function `CoT_model.generate` is invoked using a sampling-based decoding strategy (e.g., temperature sampling) to produce diverse reasoning paths across multiple runs. The final answer is extracted from each generated output and aggregated. After collecting a predefined number of samples, the frequency of each distinct answer is computed, and the most frequently occurring answer is selected, thereby implementing a majority voting mechanism.

In LLM reasoning, several methodologies have been developed to enhance models' capabilities by decomposing

Algorithm 7 Self-Consistent Predict

Require: CoT model CoT_model, Query q, number of samples $num\_samples$
Ensure: Final answer chosen by majority vote

```
answers ← []
for i = 1 to num_samples do
    output ← CoT_model.generate(q,
                                decoding = "sample")
    answer ← Extract the final answer from output
    Append answer to answers
end for
final_answer ← arg max(Counter(answers))
return final_answer
```

complex problems into intermediate steps, thereby improving both accuracy and interpretability. ToT [67] generalizes CoT by allowing models to explore multiple reasoning paths simultaneously, forming a tree-like structure. Least-to-Most Prompting [88] involves breaking down a complex problem into a sequence of subproblems arranged from the simplest to the most challenging. The model addresses each subproblem in order, using the solution of one as the foundation for the next. Zero-Shot CoT [89] enhances reasoning by appending a simple prompt such as "Let's think step by step" to the query. This minimalistic cue encourages the model to generate intermediate reasoning steps, thereby improving performance on tasks without requiring extensive prompt engineering. Program-of-Thought (PoT) [90] translates the reasoning process into a series of programming-like instructions, leveraging the structured nature of code to enforce logical consistency and precision. By framing problems in a programmatic context, models can execute complex operations systematically, which is advantageous for tasks involving mathematical computations and formal logic. Graph-of-Thoughts (GoT) [91] represents the problem-solving process as a directed acyclic graph, where nodes denote reasoning steps and edges represent dependencies. Collectively, these methods build upon the foundational principles of CoT prompting, each introducing unique structural modifications to guide models through intricate reasoning tasks. By systematically decomposing problems and exploring diverse solution pathways, these approaches significantly enhance the problem-solving capabilities of LLMs.

2) RL-Augmented Generation: RL-Augmented Generation enhances the quality of generator outputs through RL optimization [92], [93], [94], [95]. Its core paradigms can be divided into two categories: a) RLHF and b) Rule-based RL.

a) RLHF: It trains a reward model using human preference data and optimizes the language model with policy gradient algorithms. A representative method is InstructGPT [12]. Given a prompt input, and two candidate completions output_w (preferred by human annotators) and output_l (less preferred), the reward model $R_\phi(\text{input}, \text{output})$ is trained to assign higher scores to preferred outputs. To achieve this, the model estimates the probability that humans would favor output_w over output_l using a sigmoid over the reward difference:

$$P_\phi(\text{output}_w \succ \text{output}_l \mid \text{input}) = \sigma\big(R_\phi(\text{input}, \text{output}_w) - R_\phi(\text{input}, \text{output}_l)\big) \tag{43}$$

The reward model is optimized by minimizing the binary cross-entropy loss over a dataset of human comparisons

$$\mathcal{L}_{\text{RM}} = -\mathbb{E}_{(\text{input},\text{output}_w,\text{output}_l)\sim D}\big[\log \sigma\big(R_\phi(\text{input}, \text{output}_w) - R_\phi(\text{input}, \text{output}_l)\big)\big]. \tag{44}$$

Then, the policy π_θ is fine-tuned to maximize the expected reward assigned by the learned reward model. To prevent excessive divergence from the initial supervised policy π_{ref}, a KL divergence penalty is added:

$$\max_{\pi_\theta} \ \mathbb{E}_{\text{input}\sim D,\ \text{output}\sim\pi_\theta(\cdot|\text{input})}\left[R_\phi(\text{input}, \text{output})\right] - \beta\, D_{\text{KL}}\left[\pi_\theta(\cdot \mid \text{input}) \,\|\, \pi_{\text{ref}}(\cdot \mid \text{input})\right], \tag{45}$$

where β is a hyperparameter controlling the strength of the regularization. This stage is typically implemented using Proximal Policy Optimization (PPO) [59].

b) Rule-based RL: It constructs a reward function using predefined rules instead of learned neural models. A representative model is DeepSeek-R1, which applies rule-based rewards and Group Relative Policy Optimization (GRPO) to improve reasoning capabilities. Given an `input` and a generated `output`, the total reward $R(\text{input}, \text{output})$ consists of three components:

$$R(\text{input}, \text{output}) = R_{\text{correct}}(\text{output}) + R_{\text{format}}(\text{output}) + R_{\text{lang}}(\text{input}, \text{output}), \tag{46}$$

where $R_{\text{correct}}(\cdot)$ measures answer correctness, such as whether a math solution is accurate; $R_{\text{format}}(\cdot)$ ensures the reasoning process is enclosed in `<think>...</think>` tags; $R_{\text{lang}}(\text{input}, \text{output})$ checks that the output uses the same language as the input, reducing language mixing. These rewards are derived purely from deterministic rules, avoiding the use of neural reward models. Then, to optimize the language model policy $\pi_\theta(\text{output} \mid \text{input})$, DeepSeek-R1 uses GRPO, calculating a normalized advantage (A) for each sampled output in a group of size G:

$$A_i = \frac{r_i - \text{mean}(\{r_1, \ldots, r_G\})}{\text{std}(\{r_1, \ldots, r_G\})}, \tag{47}$$

where r_i is the total reward of the i-th sampled output. The objective function incorporates a clipped policy ratio and a KL divergence regularization to prevent deviation from a reference policy π_{ref}:

$$\mathcal{J}(\theta) = \mathbb{E}\left[\sum_{i=1}^{G} \text{clip}\left(\frac{\pi_\theta(\text{output}_i \mid \text{input})}{\pi_{\theta_{\text{old}}}(\text{output}_i \mid \text{input})} A_i\right)\right] - \beta D_{\text{KL}}\left[\pi_\theta(\cdot \mid \text{input}) \,\|\, \pi_{\text{ref}}(\cdot \mid \text{input})\right], \tag{48}$$

where β controls the KL penalty strength.

Compared to RLHF, which uses learned neural reward models, this approach relies entirely on rule-based signals, improving training stability and avoiding reward hacking issues. Some systems (including elements in GPT-4's safety alignment strategy) integrate rule-based reward components

TABLE II
THE COMPARISON OF DIFFERENT GENERATION METHODS.

Method		Advantages and Disadvantages
Fundamental	Encoder-Decoder	**Pros:** Clear separation of encoding and decoding. Good for conditional generation and multitask learning. Accurately incorporates input context. **Cons:** High training and inference cost. Slower generation speed. Complex architecture and tuning.
	Decoder-Only	**Pros:** Flexible autoregressive generation. Strong language modeling through pretraining. Simple architecture for fine-tuning. **Cons:** No explicit encoder limits input structure handling. Weak performance on long or structured inputs. Error accumulation in generation.
	Diffusion Models	**Pros:** Highly diverse and controllable generation. Excellent in multimodal content synthesis. Flexible style and variation control. **Cons:** Slow inference due to iterative denoising. Less mature for text generation. High training complexity and resource demand.
Advanced	Reasoning Generation	**Pros:** Enhances logical reasoning ability. Decomposes complex problems into steps. Improves the explainability of outputs. **Cons:** Generates longer and sometimes redundant outputs. Harder to control stepwise reasoning. Increased implementation and tuning complexity.
	RL-based Generation	**Pros:** Aligns outputs with user preferences or task goals. Improves control and safety of generation. Allows reward-based fine-grained supervision. **Cons:** Requires complex reward design. Often depends on human feedback. Computationally expensive and less stable to train.

to ensure outputs adhere to predefined guidelines. Such Rule-Based Reward (RBR) mechanisms offer a computationally efficient means to enforce specific behavior patterns without extensive human feedback [58].

C. Summary

Table II summarizes current generation paradigms, categorized into fundamental and advanced methods.

Among the fundamental approaches, encoder-decoder models offer a clear architectural separation between input encoding and output decoding, which is especially beneficial for conditional generation tasks such as summarization or grounded response generation. Their ability to integrate input context effectively supports multitask learning. However, the two-stage structure results in higher training and inference costs, and introduces architectural complexity. In contrast, decoder-only models rely on flexible autoregressive generation and benefit significantly from large-scale pretraining. These models are well-suited for open-ended generation tasks and creative writing. Nevertheless, the lack of an explicit encoder limits their capacity to handle long or structured inputs, and they are prone to error accumulation during generation. Diffusion models, originally developed for image synthesis, have recently been extended to multimodal generation scenarios. These models offer controllable and highly diverse outputs, making them valuable for image-text generation and creative content creation. However, their iterative denoising process incurs high inference latency, and their application to text generation remains relatively immature and resource-intensive.

TABLE III
RAG APPLICATION DOMAINS AND RELATED WORKS.

Applications	Related Work
Open-Domain QA (general knowledge)	[6], [26], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132]
Knowledge-Grounded Dialogue	[101], [109], [116], [117], [124], [127], [128], [133], [134], [135], [136]
Mathematical & Logical Reasoning	[98], [99], [108], [115], [118], [130], [132], [137]
Medical Domain Applications	[127], [130], [131], [138], [139], [140], [141], [142], [143], [144]
Legal Analysis & QA	[134], [145], [146], [147], [148], [149]
Scientific Research & Education	[112], [114], [131], [133], [134], [138], [150], [151], [152]
Fact-Checking & Verification	[6], [26], [98], [99], [100], [101], [103], [104], [105], [106], [107], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [121], [122], [123], [124], [127], [130], [131], [133], [138], [153], [152]
Multimodal Generation with Retrieval Support	[110], [121], [122], [140], [154], [155], [156], [157], [158], [159]
Code Assistants	[160], [161], [162], [163], [164], [165], [166], [167], [168], [169], [170]

Among advanced methods, reasoning-based generation explicitly guides models to decompose complex problems into intermediate reasoning steps. This significantly enhances performance in mathematical and logical reasoning tasks and improves the interpretability of outputs [96]. Yet, these methods often produce verbose outputs and are harder to control, which may hinder usability in real-world applications [97]. RL-based generation approaches introduce learning signals via human feedback or task-specific reward functions. These methods can effectively align model behavior with desired objectives such as safety, informativeness, or user preference. However, they require well-designed reward functions and high-quality training signals, which increase development complexity and computational cost.

V. APPLICATIONS

In this section, we present an overview of the major application domains of RAG, along with representative works in each domain. To provide a structured perspective, we further organize existing studies by mapping them into a method-wise matrix, where each entry corresponds to a specific combination of the aforementioned retrieval and generation strategies. This matrix captures how different combinations of retrieval and generation methods have been employed across various application settings. Finally, we summarize the state-of-the-art models for each domain to provide a comprehensive view of RAG's practical performance and deployment.

A. RAG Application Domains

Table III provides a structured overview of the application domains for RAG, along with representative related works.

Open-Domain QA (General Knowledge): This domain focuses on answering broad, fact-based questions using vast open-domain resources such as the Natural Questions dataset [171]. Due to the availability of rich data and the inherently open-ended nature of the queries, a significant volume of research has been dedicated to this area. Researchers benefit from RAG's capability to complement generation models with robust retrieval modules, enabling high-accuracy responses.

Knowledge-Grounded Dialogue: To yield responses that are well-supported by external knowledge, studies in this area use background information to generate coherent and contextually enriched dialogues [172]. The challenge lies in balancing retrieval latency with the need for conversational fluency.

Mathematical & Logical Reasoning: Applications in this domain require systems to execute multi-step reasoning and deliver precise answers with reasonable references. Due to the complexity of these tasks, the available research is relatively limited. However, when combined with specialized reasoning generation methods, RAG helps in decomposing complex problems into manageable steps.

Medical Domain Applications: Given the critical nature of medical information, RAG methods in this domain must achieve high accuracy and reliability [173]. The specialized datasets, e.g., PubMedQA [174], and the rigorous demands for safety naturally limit the volume of research, driving a focus on methods that ensure precision and domain-specific adaptation.

Legal Analysis & QA: The legal domain presents unique challenges for retrieval systems due to its requirement for fine-grained semantic understanding and precise matching of legal texts. While research in this area remains relatively limited, it typically emphasizes accuracy and interpretability, frequently employing retrieval strategies tailored to legal content.

Scientific Research & Education: This area aims at facilitating the extraction and summarization of scientific knowledge, where the integration of retrieval and generation is critical. The diversity of tasks in this domain explains the moderate level of research activity and underscores the need for multi-task and multi-domain learning approaches.

Fact-Checking & Verification: This domain is highly active, as accurate fact-checking is crucial for ensuring information reliability. In this context, RAG methods are tasked with mining relevant evidence from extensive knowledge bases, as reflected by the large number of related works. The challenge remains in preventing the propagation of retrieval errors into the final generated response.

Multimodal Generation with Retrieval Support: Addressing both textual and visual information, this domain leverages RAG's ability to handle cross-modal data. Despite the inherent complexity in aligning different modalities, the potential applications, such as image-based question answering, offer significant promise for future research.

Code Assistants: As a relatively new area, code generation systems augmented with retrieval show potential in enhancing code completion and error correction. However, the number of RAG-based studies is currently limited, probably because the integration of retrieval into code generation poses unique challenges. These include aligning natural language queries with relevant code snippets, retrieving functionally correct yet syntactically diverse examples, and effectively conditioning generation models on retrieved code under strict syntax and semantic constraints. The lack of large-scale, high-quality retrieval-augmented datasets for code further limits progress.

RAG's core advantage lies in its ability to mitigate the limitations of standalone generation models by incorporating a retrieval module to supplement factual knowledge and context. As shown in Tables I and II, each retrieval method and generation model possesses distinct strengths and weaknesses. For instance, in Open-Domain QA, where diverse and up-to-date knowledge is essential, the high recall capability of dense or hybrid retrieval methods, combined with the robust language modeling power of encoder-decoder frameworks, leads to effective performance. Conversely, in domains such as Mathematical & Logical Reasoning, the precise multi-step reasoning process is critical. Therefore, integrating specialized reasoning generation techniques becomes necessary, despite the challenges of managing longer and more complex outputs. Similarly, in safety-critical fields like Medicine and Law, the balance between retrieval precision and generation reliability is paramount. Here, targeted retrieval strategies that are tailored to domain-specific vocabularies and structured documents are essential, while the generation component must be fine-tuned to ensure adherence to factual and contextual integrity.

B. Retrieval and Generation Technique Applications

Table IV provides a structured summary of representative RAG applications, categorized by retrieval-generation combinations. It highlights which combinations are most widely studied and which are underexplored, offering insights into current research trends and design preferences across different application domains. We observe that Sparse Retrieval and Dense Retrieval dominate the landscape across nearly all generation modules, showing a significantly higher concentration of studies. This can be attributed to their maturity and accessibility: Sparse methods are simple, interpretable, and often serve as the backbone for widely used web search engines. This ubiquity in real-world applications makes Sparse Retrieval the default option in many research works, particularly in open-domain QA, dialogue systems, or fact-checking scenarios, where search engine APIs are used as retrieval modules. Dense Retrieval, by contrast, offers superior semantic matching and generalization, making it more suitable for complex tasks requiring deeper understanding, such as multi-hop reasoning, domain-specific QA, or retrieval from noisy corpora. Generative Retrieval appears less frequently in the literature (many cells marked as "NA"), and this scarcity can be explained by two factors: Generative retrievers tend to be more error-prone and lack interpretability; Generative Retrieval is used to directly generate answers from queries without the need for a separate generation module. This "retrieval-as-generation" paradigm makes traditional generator pairing unnecessary. Other retrieval methods, such as Graph, Multimodal, Hybrid, Adaptive, and Iterative Retrieval, have emerged more recently and tend to appear in more specialized tasks. For instance, Graph Retrieval is often used in scientific

TABLE IV
THE SUMMARY RAG APPLICATIONS BY DIFFERENT RETRIEVAL AND GENERATION METHODS. DIFF. DENOTES DIFFUSION MODELS.

Retriever \ Generator	Encoder-Decoder	Decoder-Only	Diff.	Reasoning Generation	RL-Augmented Generation
Sparse Retrieval	[98], [102], [116], [123], [124], [164]	[26], [98], [99], [103], [105], [111], [112], [116], [119], [120], [127], [133], [137], [138], [142], [144], [151], [153], [160], [164], [165], [166], [167]	[156]	[26], [98], [99], [112], [123], [127], [133], [137], [138], [153]	[103], [105], [111], [112], [116], [119], [120], [137], [142], [144], [160]
Dense Retrieval	[6], [100], [102], [114], [116], [117], [121], [122], [124], [128], [157], [161], [164], [169]	[101], [104], [106], [107], [108], [110], [112], [113], [114], [115], [116], [117], [118], [127], [130], [131], [132], [133], [134], [135], [138], [139], [141], [146], [147], [148], [149], [150], [152], [153], [160], [162], [163], [164], [165], [168], [170]	[154], [155], [156]	[101], [106], [107], [112], [113], [115], [118], [122], [127], [132], [133], [135], [138], [139], [146], [150], [152], [153], [170]	[101], [106], [112], [114], [115], [116], [118], [132], [135], [138], [139], [146], [152], [160], [168]
Generative Retrieval	**NA**	[136], [153]	**NA**	[153]	**NA**
Graph Retrieval	[109]	[107], [112], [129], [135], [139], [143], [146]	**NA**	[107], [112], [135], [139], [146]	[112], [135], [139], [146]
Multimodal Retrieval	[121], [122], [140], [157], [158]	[110], [159]	[154], [155]	[122], [156]	[140]
Hybrid Retrieval	[102], [124]	[107], [112], [127], [129], [133], [138], [139], [153], [165]	**NA**	[107], [112], [127], [133], [138], [139], [153]	[112], [138], [139]
Iterative Retrieval	[98], [123], [128], [158]	[26], [98], [99], [101], [106], [107], [113], [115], [118], [138], [163], [166], [168]	**NA**	[26], [98], [99], [101], [106], [107], [113], [115], [118], [123], [138]	[101], [106], [115], [118], [138], [168]
Adaptive Retrieval	[114], [121], [123], [128], [140]	[108], [114], [131], [134], [149], [167]	**NA**	[123]	[114], [140]
Post-Retrieval Augmentation	[100], [102], [117], [124]	[98], [106], [107], [111], [112], [113], [115], [117], [118], [119], [120], [133], [134], [135], [137], [138], [153]	**NA**	[98], [106], [107], [112], [113], [115], [118], [133], [135], [137], [138], [153]	[106], [111], [112], [115], [118], [119], [120], [135], [137], [138]

or biomedical domains where structured knowledge is crucial; Multimodal Retrieval appears in vision-language applications; and Hybrid or Adaptive methods are commonly introduced in resource-constrained or multi-task settings.

In terms of generation modules, we also observe clear usage patterns. First, Encoder-Decoder models are widely used, especially in combination with Sparse or Dense Retrieval, due to their structured conditional generation capability and suitability for summarization, QA, and factual generation. Then, Decoder-Only models are the foundation for open-ended generation and creative writing, offering high flexibility and transferability across tasks. Next, Diffusion Models are currently rare in text-only RAG but are emerging in multimodal tasks, where they provide controllable, iterative denoising-based generation. Next, Reasoning Generation applications are commonly found in reasoning-heavy tasks such as math and logical QA. Finally, RL-Augmented Generation applications appear in scenarios where safety, alignment, or task-specific preferences must be enforced.

By comparing Table IV with the retrieval method taxonomy (Table I), the generation method taxonomy (Table II), and the RAG application landscape (Table III), we can identify several domain-level preferences. (1) Open-domain QA, fact-checking, and knowledge-grounded dialogue favor Dense or Hybrid Retrieval, often paired with Encoder-Decoder or Decoder-Only models to balance factual grounding and fluent generation. (2) Mathematical and logical reasoning tasks require strong multi-step reasoning and factual traceability. These tasks prefer Reasoning Generation models combined with Dense or Graph Retrieval. Sparse methods are generally less suited for such tasks due to their inability to capture deep semantics and inferential relationships. (3) Medical and legal domains demand accuracy and domain specificity, thus combining Dense or Hybrid Retrieval with reliable generation models. (4) Multimodal applications pair Multimodal Retrieval with Diffusion Models or specialized Encoder-Decoder frameworks capable of processing image and text inputs. (5) Code generation typically involves retrieval of relevant snippets (often using Sparse or Dense methods) paired with Decoder-Only generation for autocomplete and task-specific generation.

C. RAG Performance Comparison on Benchmarks

To make a fair comparison between RAG and other techniques, we have selected a few representative benchmarks in major application domains. In Open-Domain QA, both the state-of-the-art model and the best RAG model are Atlas [100] with an EM of 64.00, clearly demonstrating the effectiveness of RAG in leveraging external knowledge for fact-based question answering. Conversely, in domains such as Mathematical & Logical Reasoning and Medical Domain Applications, RAG does not achieve state-of-the-art performance. In the Mathematical & Logical Reasoning domain, although PaLM 2 [177] reaches an accuracy of 90.40, the best RAG model reported (Rethinking with retrieval using GPT-3 [137]) only achieves an accuracy of 77.73. The observed performance

TABLE V
RAG APPLICATION DOMAINS, REPRESENTATIVE DATASETS, AND THE BEST PERFORMING MODELS. THE SOTA COLUMN INDICATES WHETHER RAG METHODS ACHIEVED STATE-OF-THE-ART PERFORMANCE IN THE APPLICATION DOMAIN.

Application Domain	Dataset	SOTA	SOTA Model	Best RAG Model
Open-Domain QA	Natural Questions [171]	Yes	Atlas [100] EM = 64.00	Atlas [100] EM = 64.00
Fact-Checking & Verification	KILT-FEVER [175]	Yes	Re2G [124] KILT-AC = 78.53	Re2G [124] KILT-AC = 78.53
Knowledge-Grounded Dialogue	KILT-Wizard of Wikipedia [175]	Yes	Hindsight [128] KILT-RL = 11.92	Hindsight [128] KILT-RL = 11.92
Mathematical & Logical Reasoning	StrategyQA [176]	No	PaLM 2 [177] Accuracy = 90.40	Rethinking with retrieval (GPT-3) [137] Accuracy = 77.73
Medical Domain Applications	PubMedQA [174]	No	Meditron-70B [178] Accuracy = 81.60	RankRAG-llama3-70B [127] Accuracy = =79.80
Scientific Research & Education	MMLU [179], [180]	No	DeepSeek-R1 [35] Average Accuracy = 87.50	Atlas [100] Average Accuracy = 47.90
Multimodal Generation with Retrieval Support	OK-VQA [181]	No	PaLI-X-VPD [182] Accuracy = 66.80	FLMR [157] Accuracy = 62.08

gaps are likely due to the reasoning-intensive and domain-specific nature of these tasks, which favors purely generative models. These models, equipped with internalized reasoning capabilities, are better suited to produce accurate outputs under such conditions. Similarly, in the Medical Domain, while Meditron-70B [178] (a non-RAG model) obtains an accuracy of 81.60, the best RAG model (RankRAG-llama3-70B [127]) only reaches 79.80, indicating that current RAG approaches may still struggle to fully integrate the complex, specialized medical knowledge compared to dedicated generative systems.

Furthermore, the comparison reveals that differences in retrieval strategy design play a critical role in performance variation across domains. As reflected in our earlier summary tables, systems adopting more advanced or better-aligned retrieval-generation configurations tend to achieve higher accuracy in tasks such as Open-Domain QA and Fact-Checking. In contrast, for specialized domains like scientific research or education, where the state-of-the-art models (e.g., DeepSeek-R1) achieve much higher accuracy, the best RAG models still lag behind (e.g., Atlas achieving only an average accuracy of 47.90), suggesting that further optimization, particularly in domain adaptation and retrieval integration, is needed.

In summary, while RAG frameworks have demonstrated outstanding performance in certain areas (e.g., Open-Domain QA, Fact-Checking, Knowledge-Grounded Dialogue), there remain notable challenges in other domains. The challenges include the effective integration of external retrieval with generation modules, handling domain-specific complexities, and achieving the same high performance as models that do not rely on retrieval. Future research may focus on adaptive retrieval-generation strategies, enhanced domain-specific modeling, and improved calibration between retrieved evidence and generated outputs to narrow these performance gaps.

VI. CONCLUSION

In this work, we propose a unified RAG framework built around two core modules, namely the retrieval part and the generation part. Within this framework, we formally summarize representative algorithms for both fundamental and advanced retrieval, as well as for fundamental and advanced generation techniques, using precise mathematical definitions. We also introduce the emerging use of diffusion models in RAG, which are still less explored compared to autoregressive generation methods. We then examine how modern RAG systems select and combine different retrievers and generators in practice, and provide a detailed comparison of their respective types in terms of advantages and limitations. Finally, by reviewing results on standard benchmark datasets across various application scenarios, we highlight both the strengths and remaining challenges of RAG in real-world settings.

REFERENCES

[1] R. Mao, G. Chen, X. Zhang, F. Guerin, and E. Cambria, "GPTEval: A survey on assessments of ChatGPT and GPT-4," in *Proc. LREC-COLING*. Torino, Italia: ELRA and ICCL, 2024, p. 7844–7866.

[2] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint*, 2021.

[3] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *Trans. Mach. Learn. Res.*, 2022, survey Certification.

[4] K. Du, Y. Zhao, R. Mao, F. Xing, and E. Cambria, "A retrieval-augmented multi-agent system for financial sentiment analysis," *IEEE Intell. Syst.*, vol. 40, pp. 15–22, 2025.

[5] A. Roberts, C. Raffel, and N. Shazeer, "How much knowledge can you pack into the parameters of a language model?" in *Proc. EMNLP*, 2020, pp. 5418–5426.

[6] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *NeurIPS*, vol. 33, pp. 9459–9474, 2020.

[7] D. Cai, Y. Wang, L. Liu, and S. Shi, "Recent advances in retrieval-augmented text generation," in *Proc. SIGIR*, 2022, pp. 3417–3419.

[8] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint*, vol. 2, p. 1, 2023.

[9] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T.-S. Chua, and Q. Li, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. SIGKDD*, 2024, pp. 6491–6501.

[10] P. Zhao, H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, and B. Cui, "Retrieval-augmented generation for AI-generated content: A survey," *arXiv preprint*, 2024.

[11] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *NeurIPS*, vol. 35, pp. 24 824–24 837, 2022.

[12] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin *et al.*, "Training language models to follow instructions with human feedback," *NeurIPS*, vol. 35, pp. 27 730–27 744, 2022.

[13] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: BM25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, no. 4, pp. 333–389, 2009.

[14] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Inf. Process. Manag.*, vol. 24, no. 5, pp. 513–523, 1988.

[15] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense passage retrieval for open-domain question answering," in *Proc. EMNLP*, 2020, pp. 6769–6781.

[16] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-Sensitive hashing scheme based on P-Stable distributions," in *Proc. SCG*, 2004, pp. 253–262.

[17] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The FAISS library," *arXiv preprint*, 2024.

[18] Y. Tay, V. Tran, M. Dehghani, J. Ni, D. Bahri, H. Mehta, Z. Qin, K. Hui, Z. Zhao, J. Gupta *et al.*, "Transformer memory as a differentiable search index," *NeurIPS*, vol. 35, pp. 21 831–21 843, 2022.

[19] N. De Cao, G. Izacard, S. Riedel, and F. Petroni, "Autoregressive entity retrieval," in *Proc. ICLR*, 2021.

[20] X. He, Y. Tian, Y. Sun, N. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, "G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering," *NeurIPS*, vol. 37, pp. 132 876–132 907, 2024.

[21] C. Mavromatis and G. Karypis, "GNN-RAG: Graph neural retrieval for large language model reasoning," *arXiv preprint*, 2024.

[22] C. Wei, Y. Chen, H. Chen, H. Hu, G. Zhang, J. Fu, A. Ritter, and W. Chen, "UniIR: Training and benchmarking universal multimodal information retrievers," in *Proc. ECCV*, 2024, pp. 387–404.

[23] J. A. Portillo-Quintero, J. C. Ortiz-Bayliss, and H. Terashima-Marín, "A straightforward framework for video retrieval using CLIP," in *Proc. MCPR*, 2021, pp. 3–12.

[24] B. Zhu, C.-W. Ngo, J. Chen, and Y. Hao, "R2GAN: Cross-modal recipe retrieval with generative adversarial network," in *Proc. CVPR*, 2019, pp. 11 477–11 486.

[25] G. V. Cormack, C. L. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proc. SIGIR*, 2009, pp. 758–759.

[26] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. ICLR*, 2023.

[27] X. Tang, Q. Gao, J. Li, N. Du, Q. Li, and S. Xie, "MBA-RAG: A bandit approach for adaptive retrieval-augmented generation through question complexity," in *Proc. COLING*, 2025, pp. 3248–3254.

[28] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *arXiv preprint*, 2019.

[29] Z. Qiao, W. Ye, Y. Jiang, T. Mo, P. Xie, W. Li, F. Huang, and S. Zhang, "Supportiveness-based knowledge rewriting for retrieval-augmented language modeling," in *Find. NAACL*, 2025, pp. 2728–2740.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[31] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. ACL*, 2020, pp. 7871–7880.

[32] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," *OpenAI Tech Report*, 2018.

[33] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "PaLM: Scaling language modeling with pathways," *J. Mach. Learn. Res.*, vol. 24, no. 240, pp. 1–113, 2023.

[34] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *NeurIPS*, vol. 33, pp. 6840–6851, 2020.

[35] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning," *arXiv preprint*, 2025.

[36] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *J. Doc.*, vol. 28, no. 1, pp. 11–21, 1972.

[37] Q. Liu, R. Mao, X. Geng, and E. Cambria, "Semantic matching in machine reading comprehension: An empirical study," *Inf. Process. Manag.*, vol. 60, no. 2, p. 103145, 2023.

[38] M. Oh, J. Lee, J. Li, and G. Wang, "PK-ICR: Persona-knowledge interactive multi-context retrieval for grounded dialogue," in *Proc. EMNLP*, 2023, pp. 16 383–16 395.

[39] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019, pp. 4171–4186.

[40] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2010.

[41] H. Liu, S. Huang, J. Liu, Y. Zhan, H. Sun, W. Deng, F. Sun, F. Wei, and Q. Zhang, "GeAR: Generation augmented retrieval," *arXiv preprint*, 2025.

[42] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[43] Q. Lin, R. Mao, J. Liu, F. Xu, and E. Cambria, "Fusing topology contexts and logical rules in language models for knowledge graph completion," *Inf. Fusion*, vol. 90, pp. 253–264, 2023.

[44] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering," in *Proc. NAACL*, 2021, pp. 535–546.

[45] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. ICML*, 2021, pp. 8748–8763.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[47] D. Lee, S.-w. Hwang, K. Lee, S. Choi, and S. Park, "On complementarity objectives for hybrid retrieval," in *Proc. ACL*, 2023, pp. 13 357–13 368.

[48] J. Mu, X. Li, and N. Goodman, "Learning to compress prompts with gist tokens," *NeurIPS*, vol. 36, pp. 19 327–19 352, 2023.

[49] A. Chevalier, A. Wettig, A. Ajith, and D. Chen, "Adapting language models to compress contexts," in *Proc. EMNLP*, 2023.

[50] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep., 1999.

[51] N. Arabzadeh, X. Yan, and C. L. Clarke, "Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection," in *Proc. CIKM*, 2021, pp. 2862–2866.

[52] W. X. Zhao, J. Liu, R. Ren, and J.-R. Wen, "Dense text retrieval based on pretrained language models: A survey," *ACM Trans. Inf. Syst.*, vol. 42, no. 4, pp. 1–60, 2024.

[53] X. Li, J. Jin, Y. Zhou, Y. Zhang, P. Zhang, Y. Zhu, and Z. Dou, "From matching to generation: A survey on generative information retrieval," *ACM Trans. Inf. Syst.*, 2024.

[54] T. Shen, R. Mao, J. Wang, X. Zhang, and E. Cambria, "Flow-guided direct preference optimization for knowledge graph reasoning with trees," in *Proc. SIGIR*, 2025.

[55] T. Wang, F. Li, L. Zhu, J. Li, Z. Zhang, and H. T. Shen, "Cross-modal retrieval: a systematic review of methods and future directions," *Proc. IEEE*, 2025.

[56] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[57] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *NeurIPS*, vol. 33, pp. 1877–1901, 2020.

[58] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "GPT-4 technical report," *arXiv preprint*, 2023.

[59] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Roziere, N. Goyal, E. Hambro, F. Azhar *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint*, 2023.

[60] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint*, 2023.

[61] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint*, 2024.

[62] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. d. l. Casas, E. B. Hanna, F. Bressand *et al.*, "Mixtral of experts," *arXiv preprint*, 2024.

[63] A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan *et al.*, "DeepSeek-V3 technical report," *arXiv preprint*, 2024.

[64] D. Guo, Q. Zhu, D. Yang, Z. Xie, K. Dong, W. Zhang, G. Chen, X. Bi, Y. Wu, Y. Li *et al.*, "DeepSeek-Coder: When the large language model meets programming-the rise of code intelligence," *CoRR*, 2024.

[65] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu *et al.*, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv preprint*, 2024.

[66] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. CVPR*, 2022, pp. 10 684–10 695.

[67] S. Yao, D. Yu, J. Zhao, I. Shafran, T. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *NeurIPS*, vol. 36, pp. 11 809–11 822, 2023.

[68] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence-to-sequence learning with neural networks," *NeurIPS*, vol. 27, 2014.

[69] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[70] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[71] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence-to-sequence learning," in *Proc. ICML*, 2017, pp. 1243–1252.

[72] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *Proc. ICLR*, 2021.

[73] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," *NeurIPS*, vol. 34, pp. 8780–8794, 2021.

[74] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever, "Consistency models," in *Proc. ICML*, 2023, pp. 32 211–32 252.

[75] S. Chen, M. Xu, J. Ren, Y. Cong, S. He, Y. Xie, A. Sinha, P. Luo, T. Xiang, and J.-M. Perez-Rua, "Gentron: Diffusion transformers for image and video generation," in *Proc. CVPR*, 2024, pp. 6441–6451.

[76] T. Karras, M. Aittala, J. Lehtinen, J. Hellsten, T. Aila, and S. Laine, "Analyzing and improving the training dynamics of diffusion models," in *Proc. CVPR*, 2024, pp. 24 174–24 184.

[77] Z. Kadkhodaie, F. Guth, E. P. Simoncelli, and S. Mallat, "Generalization in diffusion models arises from geometry-adaptive harmonic representations," in *Proc. ICLR*, 2024.

[78] T. Karras, M. Aittala, T. Kynkäänniemi, J. Lehtinen, T. Aila, and S. Laine, "Guiding a diffusion model with a bad version of itself," *NeurIPS*, vol. 37, pp. 52 996–53 021, 2024.

[79] D. Zhang, J. Wang, and F. Luo, "Directly denoising diffusion models," in *Proc. ICML*, 2024, pp. 59 951–59 974.

[80] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "WaveGrad: Estimating gradients for waveform generation," in *Proc. ICLR*, 2021.

[81] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *Proc. ICLR*, 2021.

[82] L. Zhou, Y. Du, and J. Wu, "3D shape generation and completion through point-voxel diffusion," in *Proc. ICCV*, 2021, pp. 5826–5835.

[83] N. Anand and T. Achim, "Protein structure and sequence generation with equivariant denoising diffusion probabilistic models," *arXiv preprint*, 2022.

[84] Q. Lin, K. He, Y. Zhu, F. Xu, E. Cambria, and M. Feng, "Cross-modal knowledge diffusion-based generation for difference-aware medical VQA," *IEEE Trans. Image Process.*, 2025.

[85] F. Xu, Q. Hao, Z. Zong, J. Wang, Y. Zhang, J. Wang, X. Lan, J. Gong, T. Ouyang, F. Meng *et al.*, "Towards large reasoning models: A survey of reinforced reasoning with large language models," *arXiv preprint*, 2025.

[86] R. Mao, Q. Liu, K. He, W. Li, and E. Cambria, "The biases of pre-trained language models: An empirical study on prompt-based sentiment analysis and emotion detection," *IEEE Trans. Affect. Comput.*, vol. 14, no. 3, pp. 1743–1753, 2023.

[87] X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," in *Proc. ICLR*, 2023.

[88] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. V. Le, and E. H. Chi, "Least-to-most prompting enables complex reasoning in large language models," in *Proc. ICLR*, 2023.

[89] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are Zero-Shot reasoners," *NeurIPS*, vol. 35, pp. 22 199–22 213, 2022.

[90] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *Trans. Mach. Learn. Res.*, 2023.

[91] M. Besta, N. Blach, A. Kubicek, R. Gerstenberger, M. Podstawski, L. Gianinazzi, J. Gajda, T. Lehmann, H. Niewiadomski, P. Nyczyk *et al.*, "Graph of thoughts: Solving elaborate problems with large language models," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 16, 2024, pp. 17 682–17 690.

[92] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT Press, Cambridge, 1998, vol. 1, no. 1.

[93] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, "Learning to summarize with human feedback," *NeurIPS*, vol. 33, pp. 3008–3021, 2020.

[94] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *NeurIPS*, vol. 30, 2017.

[95] X. Wu, "Sailing AI by the stars: A survey of learning from rewards in post-training and test-time scaling of large language models," *arXiv preprint*, 2025.

[96] Z. Chu, J. Chen, Q. Chen, W. Yu, T. He, H. Wang, W. Peng, M. Liu, B. Qin, and T. Liu, "Navigate through enigmatic labyrinth: a survey of chain of thought reasoning: advances, frontiers and future," in *Proc. ACL*, 2024, pp. 1173–1203.

[97] E. Cambria, R. Mao, M. Chen, Z. Wang, and S.-B. Ho, "Seven pillars for the future of artificial intelligence," *IEEE Intell. Syst.*, vol. 38, no. 6, pp. 62–69, 2023.

[98] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," in *Proc. ACL*, 2023, pp. 10 014–10 037.

[99] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith, and M. Lewis, "Measuring and narrowing the compositionality gap in language models," in *Proc. EMNLP*, 2023.

[100] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "ATLAS: Few-Shot learning with retrieval augmented language models," *J. Mach. Learn. Res.*, vol. 24, no. 251, pp. 1–43, 2023.

[101] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia, "Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP," *arXiv preprint*, 2022.

[102] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open-domain question answering," in *Proc. EACL*, 2021, pp. 874–880.

[103] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, "Query rewriting in retrieval-augmented large language models," in *Proc. EMNLP*, 2023, pp. 5303–5315.

[104] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. B. Van Den Driessche, J.-B. Lespiau, B. Damoc, A. Clark *et al.*, "Improving language models by retrieving from trillions of tokens," in *Proc. ICML*, 2022, pp. 2206–2240.

[105] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders *et al.*, "WebGPT: Browser-assisted question-answering with human feedback," *arXiv preprint*, 2021.

[106] Y. Chen, L. Yan, W. Sun, X. Ma, Y. Zhang, S. Wang, D. Yin, Y. Yang, and J. Mao, "Improving retrieval-augmented generation through multi-agent reinforcement learning," *arXiv preprint*, 2025.

[107] Z. Shen, C. Diao, P. Vougiouklis, P. Merita, S. Piramanayagam, D. Graux, D. Tu, Z. Jiang, R. Lai, Y. Ren *et al.*, "GeAR: Graph-enhanced agent for retrieval-augmented generation," *arXiv preprint*, 2024.

[108] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. tau Yih, "REPLUG: Retrieval-augmented black-box language models," in *Proc. NAACL*, 2024, pp. 8371–8384.

[109] M. Kang, J. M. Kwak, J. Baek, and S. J. Hwang, "Knowledge graph-augmented language models for knowledge-grounded dialogue generation," *arXiv preprint*, 2023.

[110] M. Yasunaga, A. Aghajanyan, W. Shi, R. James, J. Leskovec, P. Liang, M. Lewis, L. Zettlemoyer, and W.-T. Yih, "Retrieval-augmented multimodal language modeling," in *Proc. ICML*, 2023, pp. 39 755–39 769.

[111] J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, F. Song, M. Chadwick, M. Glaese, S. Young, L. Campbell-Gillingham, G. Irving *et al.*, "Teaching language models to support answers with verified quotes," *arXiv preprint*, 2022.

[112] Y. Yuan, L. Chengwu, J. Yuan, G. Sun, S. Li, and M. Zhang, "A hybrid RAG system with comprehensive enhancement on complex reasoning," in *Proc. KDD Cup Workshop*, 2024.

[113] T. Yu, S. Zhang, and Y. Feng, "Auto-RAG: Autonomous retrieval-augmented generation for large language models," *arXiv preprint*, 2024.

[114] Z. Yu, C. Xiong, S. Yu, and Z. Liu, "Augmentation-adapted retriever improves generalization of language models as generic plug-in," in *Proc. ACL*, 2023.

[115] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," in *Find. EMNLP*, 2023, pp. 9248–9274.

[116] H. Yang, Z. Li, Y. Zhang, J. Wang, N. Cheng, M. Li, and J. Xiao, "PRCA: Fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter," in *Proc. EMNLP*, 2023, pp. 5364–5375.

[117] Z. Wang, J. Araki, Z. Jiang, M. R. Parvez, and G. Neubig, "Learning to filter context for retrieval-augmented generation," *arXiv preprint*, 2023.

[118] S. Xu, L. Pang, H. Shen, X. Cheng, and T.-S. Chua, "Search-in-the-Chain: Interactively enhancing large language models with search for knowledge-intensive tasks," in *Proc. WebConf*, 2024, pp. 1362–1373.

[119] S. Mao, Y. Jiang, B. Chen, X. Li, P. Wang, X. Wang, P. Xie, F. Huang, H. Chen, and N. Zhang, "RaFe: Ranking feedback improves query rewriting for RAG," in *Find. EMNLP*, 2024, pp. 884–901.

[120] Z. Li, J. Wang, Z. Jiang, H. Mao, Z. Chen, J. Du, Y. Zhang, F. Zhang, D. Zhang, and Y. Liu, "DMQR-RAG: Diverse multi-query rewriting for RAG," *arXiv preprint*, 2024.

[121] Z. Hu, A. Iscen, C. Sun, Z. Wang, K.-W. Chang, Y. Sun, C. Schmid, D. A. Ross, and A. Fathi, "REVEAL: Retrieval-augmented visual-language pre-training with multi-source multimodal knowledge memory," in *Proc. CVPR*, 2023, pp. 23 369–23 379.

[122] W. Chen, H. Hu, X. Chen, P. Verga, and W. Cohen, "MuRAG: Multimodal retrieval-augmented generator for open question answering over images and text," in *Proc. EMNLP*, 2022, pp. 5558–5570.

[123] S. Jeong, J. Baek, S. Cho, S. J. Hwang, and J. C. Park, "Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity," in *Proc. NAACL*, 2024, pp. 7029–7043.

[124] M. Glass, G. Rossiello, M. F. M. Chowdhury, A. Naik, P. Cai, and A. Gliozzo, "Re2G: Retrieve, rerank, generate," in *Proc. NAACL*, 2022, pp. 2701–2715.

[125] X. Wu, L. Pan, W. Y. Wang, and A. T. Luu, "AKEW: Assessing knowledge editing in the wild," in *Proc. EMNLP*, Nov. 2024, pp. 15 118–15 133.

[126] X. Wu, L. Pan, Y. Xie, R. Zhou, S. Zhao, Y. Ma, M. Du, R. Mao, A. T. Luu, and W. Y. Wang, "AntiLeak-Bench: Preventing data contamination by automatically constructing benchmarks with updated real-world knowledge," *arXiv preprint*, 2024.

[127] Y. Yu, W. Ping, Z. Liu, B. Wang, J. You, C. Zhang, M. Shoeybi, and B. Catanzaro, "RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs," *NeurIPS*, vol. 37, pp. 121 156–121 184, 2024.

[128] A. Paranjape, O. Khattab, C. Potts, M. Zaharia, and C. D. Manning, "Hindsight: Posterior-guided training of retrievers for improved open-ended generation," in *Proc. ICLR*, 2021.

[129] B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "HippoRAG: Neurobiologically inspired long-term memory for large language models," in *NeurIPS*, 2024.

[130] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proc. ICLR*, 2023.

[131] S. Islam, M. A. Rahman, K. T. Hossain, E. Hoque, S. Joty, and M. R. Parvez, "Open-RAG: Enhanced retrieval augmented reasoning with open-source large language models," in *Find. EMNLP*, 2024, pp. 14 231–14 244.

[132] J. Wang, M. Chen, B. Hu, D. Yang, Z. Liu, Y. Shen, P. Wei, Z. Zhang, J. Gu, J. Zhou *et al.*, "Learning to plan for retrieval-augmented large language models from knowledge graphs," in *Find. EMNLP*, 2024, pp. 7813–7835.

[133] L. S. Nguyen and T. T. Quan, "URAG: Implementing a unified hybrid RAG for precise answers in university admission chatbots – a case study at HCMUT," in *Proc. ISICT*, 2024, pp. 82–93.

[134] Z. Wang, S. Teo, J. Ouyang, Y. Xu, and W. Shi, "M-RAG: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions," in *Proc. ACL*, 2024, pp. 1966–1978.

[135] J. Yu, S. Wu, J. Chen, and W. Zhou, "LLMs as collaborator: Demands-guided collaborative retrieval-augmented generation for commonsense knowledge-grounded open-domain dialogue systems," in *Find. EMNLP*, 2024, pp. 13 586–13 612.

[136] F. Pallucchini, X. Zhang, R. Mao, and E. Cambria, "Self-explanatory and retrieval-augmented LLMs for financial sentiment analysis," in *ACM/SIGAPP SAC*, Catania, Italy, 2025.

[137] H. He, H. Zhang, and D. Roth, "Rethinking with retrieval: Faithful large language model inference," *arXiv preprint*, 2022.

[138] J. Lála, O. O'Donoghue, A. Shtedritski, S. Cox, S. G. Rodriques, and A. D. White, "PaperQA: Retrieval-augmented generative agent for scientific research," *arXiv preprint*, 2023.

[139] X. Zhao, S. Liu, S.-Y. Yang, and C. Miao, "MedRAG: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot," in *Proc. WebConf*, 2025.

[140] P. Xia, K. Zhu, H. Li, T. Wang, W. Shi, S. Wang, L. Zhang, J. Zou, and H. Yao, "MMed-RAG: Versatile multimodal RAG system for medical vision language models," in *Proc. NeurIPS Safe Generative AI Workshop*, 2024.

[141] C. S. Ong, N. T. Obey, Y. Zheng, A. Cohan, and E. B. Schneider, "SurgeryLLM: A retrieval-augmented generation large language model framework for surgical decision support and workflow enhancement," *npj Digit. Med.*, vol. 7, no. 1, p. 364, 2024.

[142] T. K. Hung, G. J. Kuperman, E. J. Sherman, A. L. Ho, C. Weng, D. G. Pfister, and J. J. Mao, "Performance of retrieval-augmented large language models to recommend head and neck cancer clinical trials," *J. Med. Internet Res.*, vol. 26, p. e60695, 2024.

[143] J. Wu, J. Zhu, and Y. Qi, "Medical graph RAG: Towards safe medical large language model via graph retrieval-augmented generation," *CoRR*, 2024.

[144] A. M. García, D. Benavent, B. M. Barbancho, and D. F. Núñez, "Optimizing the clinical application of rheumatology guidelines using large language models: A retrieval-augmented generation framework integrating acr and eular recommendations," *medRxiv preprint*, 2025.

[145] N. Pipitone and G. H. Alami, "LegalBench-RAG: A benchmark for retrieval-augmented generation in the legal domain," *arXiv preprint*, 2024.

[146] N. H. Thanh and K. Satoh, "KRAG framework for enhancing llms in the legal domain," *arXiv preprint*, 2024.

[147] A. Louis, G. van Dijck, and G. Spanakis, "Interpretable long-form legal question answering with retrieval-augmented large language models," in *Proc. AAAI*, vol. 38, no. 20, 2024, pp. 22 266–22 275.

[148] N. Wiratunga, R. Abeyratne, L. Jayawardena, K. Martin, S. Massie, I. Nkisi-Orji, R. Weerasinghe, A. Liret, and B. Fleisch, "CBR-RAG: Case-based reasoning for retrieval augmented generation in llms for legal question answering," in *Proc. ICCBR*, 2024, pp. 445–460.

[149] Y. Zhang, D. Li, G. Peng, S. Guo, Y. Dou, and R. Yi, "A dynamic retrieval-augmented generation framework for border inspection legal question answering," in *Proc. IALP*, 2024, pp. 372–376.

[150] M. H. Prince, H. Chan, A. Vriza, T. Zhou, V. K. Sastry, Y. Luo, M. T. Dearing, R. J. Harder, R. K. Vasudevan, and M. J. Cherukara, "Opportunities for retrieval and tool augmented large language models in scientific facilities," *npj Comput. Mater.*, vol. 10, no. 1, p. 251, 2024.

[151] M. P. Polak and D. Morgan, "Extracting accurate materials data from research papers with conversational language models and prompt engineering," *Nat. Commun.*, vol. 15, no. 1, p. 1569, 2024.

[152] M. Leippold, S. A. Vaghefi, D. Stammbach, V. Muccione, J. Bingler, J. Ni, C. C. Senni, T. Wekhof, T. Schimanski, G. Gostlow *et al.*, "Automated fact-checking of climate claims with large language models," *npj Clim. Action*, vol. 4, no. 1, p. 17, 2025.

[153] Y. Yoon, J. Jung, S. Yoon, and K. Park, "HerO at AVeriTeC: The herd of open large language models for verifying real-world claims," in *Proc. FEVER*, 2024, pp. 130–136.

[154] A. Blattmann, R. Rombach, K. Oktay, J. Müller, and B. Ommer, "Retrieval-augmented diffusion models," *NeurIPS*, vol. 35, pp. 15 309–15 324, 2022.

[155] M. Zhang, X. Guo, L. Pan, Z. Cai, F. Hong, H. Li, L. Yang, and Z. Liu, "Remodiffuse: Retrieval-augmented motion diffusion model," in *Proc. ICCV*, 2023, pp. 364–373.

[156] W. Chen, H. Hu, C. Saharia, and W. W. Cohen, "Re-Imagen: Retrieval-augmented text-to-image generator," in *Proc. ICLR*, 2022.

[157] W. Lin, J. Chen, J. Mei, A. Coca, and B. Byrne, "Fine-grained late-interaction multi-modal retrieval for retrieval-augmented visual question answering," *NeurIPS*, vol. 36, pp. 22 820–22 840, 2023.

[158] O. Adjali, O. Ferret, S. Ghannay, and H. Le Borgne, "Multi-level information retrieval augmented generation for knowledge-based visual question answering," in *Proc. EMNLP*, 2024, pp. 16 499–16 513.

[159] M. A. Arefeen, B. Debnath, M. Y. S. Uddin, and S. Chakradhar, "iRAG: Advancing RAG for videos with an incremental approach," in *Proc. CIKM*, 2024, pp. 4341–4348.

[160] N. Nashid, M. Sintaha, and A. Mesbah, "Retrieval-based prompt selection for code-related few-shot learning," in *Proc. ICSE*, 2023, pp. 2450–2462.
[161] H. Wang, X. Xia, D. Lo, Q. He, X. Wang, and J. Grundy, "Context-aware retrieval-based deep commit message generation," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 4, pp. 1–30, 2021.
[162] H. Koziolek, S. Grüner, R. Hark, V. Ashiwal, S. Linsbauer, and N. Eskandani, "LLM-based and retrieval-augmented control code generation," in *Proc. LLM4Code*, 2024, pp. 22–29.
[163] H. Su, S. Jiang, Y. Lai, H. Wu, B. Shi, C. Liu, Q. Liu, and T. Yu, "EvoR: Evolving retrieval for code generation," in *Find. EMNLP*, 2024, pp. 2538–2554.
[164] S. Zhou, U. Alon, F. F. Xu, Z. Jiang, and G. Neubig, "DocPrompting: Generating code by retrieving the docs," in *Proc. ICLR*, 2023.
[165] X. Gao, Y. Xiong, D. Wang, Z. Guan, Z. Shi, H. Wang, and S. Li, "Preference-Guided refactored tuning for retrieval-augmented code generation," in *Proc. ASE*, 2024, pp. 65–77.
[166] F. Zhang, B. Chen, Y. Zhang, J. Keung, J. Liu, D. Zan, Y. Mao, J.-G. Lou, and W. Chen, "RepoCoder: Repository-level code completion through iterative retrieval and generation," in *Proc. EMNLP*, 2023, pp. 2471–2484.
[167] D. Wu, W. U. Ahmad, D. Zhang, M. K. Ramanathan, and X. Ma, "REPOFORMER: Selective retrieval for repository-level code completion," in *Proc. ICML*, 2024, pp. 53 270–53 290.
[168] A. Dutta, M. Singh, G. Verbruggen, S. Gulwani, and V. Le, "RAR: Retrieval-augmented retrieval for code generation in low-resource languages," in *Proc. EMNLP*, 2024, pp. 21 506–21 515.
[169] X. Zhang, Y. Zhou, G. Yang, and T. Chen, "Syntax-aware retrieval-augmented code generation," in *Find. EMNLP*, 2023, pp. 1291–1302.
[170] J. Yoo, H. Han, Y. Lee, J. Kim, and S.-w. Hwang, "PERC: Plan-as-query example retrieval for underrepresented code generation," in *Proc. COLING*, 2025, pp. 7982–7997.
[171] C. Alberti, K. Lee, and M. Collins, "A BERT baseline for the natural questions," *arXiv preprint*, 2019.
[172] X. Lan, F. Wu, K. He, Q. Zhao, S. Hong, and M. Feng, "GEM: Empowering MLLM for grounded ECG understanding with time series and images," *arXiv preprint*, 2025.
[173] K. He, R. Mao, Q. Lin, Y. Ruan, X. Lan, M. Feng, and E. Cambria, "A survey of large language models for healthcare: from data, technology, and applications to accountability and ethics," *Inf. Fusion*, p. 102963, 2025.
[174] Q. Jin, B. Dhingra, Z. Liu, W. Cohen, and X. Lu, "PubMedQA: A dataset for biomedical research question answering," in *Proc. EMNLP-IJCNLP*, 2019, pp. 2567–2577.
[175] F. Petroni, A. Piktus, A. Fan, P. Lewis, M. Yazdani, N. De Cao, J. Thorne, Y. Jernite, V. Karpukhin, J. Maillard *et al.*, "KILT: A benchmark for knowledge intensive language tasks," in *Proc. NAACL*, 2021, pp. 2523–2544.
[176] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant, "Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies," *Trans. Assoc. Comput. Linguist.*, vol. 9, pp. 346–361, 2021.
[177] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen *et al.*, "PaLM 2 technical report," *arXiv preprint*, 2023.
[178] Z. Chen, A. H. Cano, A. Romanou, A. Bonnet, K. Matoba, F. Salvi, M. Pagliardini, S. Fan, A. Kopf, A. Mohtashami *et al.*, "Meditron-70B: Scaling medical pretraining for large language models," *arXiv preprint*, 2023.
[179] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," in *Proc. ICLR*, 2021.
[180] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "Aligning AI with shared human values," in *Proc. ICLR*, 2021.
[181] K. Marino, M. Rastegari, A. Farhadi, and R. Mottaghi, "OK-VQA: A visual question answering benchmark requiring external knowledge," in *Proc. CVPR*, 2019, pp. 3195–3204.
[182] Y. Hu, O. Stretcu, C.-T. Lu, K. Viswanathan, K. Hata, E. Luo, R. Krishna, and A. Fuxman, "Visual program distillation: Distilling tools and programmatic reasoning into vision-language models," in *Proc. CVPR*, 2024, pp. 9590–9601.

Zihao Huang is a PhD candidate at Nanyang Technological University, supervised by Prof. Erik Cambria and Dr. Rui Mao. He obtained his B.Eng. degree in Computer Science and Technology from Hunan University, and MSc in Artificial Intelligence from the National University of Singapore. His research interests include NLP, information retrieval, large language models, and their applications in finance. Contact him at littleinorganic@gmail.com.

Rui Mao is a Research Scientist and Lead Investigator at Nanyang Technological University. He obtained his Ph.D. degree in Computing Science from the University of Aberdeen. His research interest lies in NLP, cognitive computing, and their applications in finance and cognitive science. He contributes to the scholarly community as an associate editor for journals such as IEEE Transactions on Affective Computing, and Information Fusion. Contact him at rui.mao@ntu.edu.sg.

Xiaobao Wu is a Research Scientist at Nanyang Technological University. He obtained his Ph.D. degree from Nanyang Technological University. His research interest lies in NLP and AI, especially in reasoning, trustworthy, and safety of large language models. He has published over 30 papers in top conferences and journals like ICML, NeurIPS, AAAI, ACL, and EMNLP. He also served as Area Chair in ACL. Contact him at xiaobao.wu@ntu.edu.sg.

Kai He earned his doctorate from Xi'an Jiaotong University under the supervision of Professor Li Chen. He also completed an academic visit at Nanyang Technological University, under the guidance of Professor Erik Cambria (IEEE Fellow). Currently, he is a postdoctoral researcher at the School of Public Health, National University of Singapore, focusing on research in Large Language Model, AI for Healthcare, Affective Computing, Information Extraction. Contact him at kai_he@nus.edu.sg.

Xulang Zhang is a Research Fellow at Nanyang Technological University. She obtained her Ph.D. degree in Computing and Data Science from Nanyang Technological University under the supervision of Professor Erik Cambria, focusing on neurosymbolic sentiment analysis. Her current research interests include bias analysis, affective computing, and explainability in natural language processing. Contact her at xulang.zhang@ntu.edu.sg.

Erik Cambria is a Professor at Nanyang Technological University, where he also holds the appointment of Provost Chair in Computer Science and Engineering. His research focuses on neurosymbolic AI for trustworthy and explainable affective computing in social media monitoring, financial forecasting, and AI for social good. He is an IEEE Fellow, Associate Editor of various top-tier AI journals, and is involved in several international conferences as a keynote speaker, program chair, and committee member. Contact him at cambria@ntu.edu.sg.