A Case Study on Retrieval-Augmented Generation for AI-Generated Content

Aditya Kumar
University Institute of Engineering
Chandigarh University
Mohali, India
24mai14003@cuchd.in

***Abstract*—Improvements in model algorithms has led to the development of Artificial Intelligence Generated Content (AIGC), aided by the expansion of core models, and the availability of high-quality datasets. Even with its noteworthy accomplishments, AIGC still confronts challenges that include keeping up with new information, managing large amounts of training and inference data, minimizing data leakage, and handling long tail data. The paradigm known as Retrieval-Augmented Generation (RAG) has now surfaced as a solution to these problems. Specifically, RAG presents the information retrieval procedure that improves the generation process by obtaining pertinent objects from accessible data sources, resulting in increased robustness and accuracy. In this study, we present a thorough overview of previous attempts to include RAG methodologies into AIGC scenarios. In order to isolate the essential abstractions of the augmentation approaches for different retrievers and generators, we first categorize RAG foundations based on how the retriever augments the generator. All RAG situations are covered by this cohesive viewpoint, which highlights developments and important technology that support possible future breakthroughs. We also provide a summary of further RAG enhancement techniques that help with efficient RAG system deployment and engineering. Then, looking at things from a different angle, we survey on real-world RAG applications across many modalities and tasks, providing insightful references for scholars and professionals. We also go over the shortcomings of the existing RAG systems, present the benchmarks for RAG, and make some recommendations for future research paths.**

***Index Terms*—Retrieval-augmented generation, AI-generated content, generative models, information retrieval.**

I. Introduction

A. Background

Artificial Intelligence Generated Content (AIGC) has seen a surge in attention in recent years. Large Language Models (LLMs) such as the GPT series [1]–[3] and the LLAMA series [4]–[6] for texts and codes, DALLE [7]–[9] and Stable Diffusion [10] for images, and Sora [11] novel model algorithms, explosive scale of foundation models, and massive high-quality datasets for videos are just a few examples of the carefully designed content generation tools that can produce a wide range of outputs across different modalities. The term "AIGC" highlights that sophisticated generative models, instead of humans or rule-based methods, are used to construct the contents. Consequently, the use of cutting-edge model methods, enormous, high-quality datasets, and foundation models with an exponential scale, these generative models have demonstrated outstanding performance. In particular, image-generation tasks have moved from Generative Adversarial Networks (GANs) [12] to Latent Diffusion Models (LDMs) [10], while sequence-to-sequence tasks have moved from using Long Short Term Memory (LSTM) networks [13] to Transformer-based models [14]. It is noteworthy that the architecture of foundation models has expanded from millions of parameters at first [15], [16], to billions or even trillions of parameters at this point [1], [4], [17]. The availability of extensive, high-quality datasets [1], [18], which offer enough training examples to completely tune model parameters, further supports these developments.

Another essential use in computer science is information retrieval. Retrieval seeks to identify pertinent things that already exist from a sizable pool of resources, as compared to generation. Web search engines, which are primarily concerned with document retrieval, are the most common applications of retrieval [19], [20]. Currently, billion-scale document collections can be handled by effective information retrieval systems [21], [22]. Retrieval has been used for many additional modalities in addition to documents [23]–[26].

Even with major progress in generative models, AIGC still faces obstacles like as out-of-date knowledge, a lack of long-tail knowledge [27], and the possibility of private training data leaks [28]. Retrieval-Augmented Generation (RAG) uses a flexible data store to try to alleviate these problems [29]. Retrievable knowledge serves as non-parametric memory that may encode sensitive information, is readily updated, and can handle a large amount of long-tail knowledge. Retrieval can also reduce the cost of generation. Large models can be made smaller with RAG [30], extended contexts can be supported [31], and certain generation processes can be removed [32].

The retriever receives an input query, finds pertinent data sources, and uses the knowledge to better the generating process by interacting with the generator. Depending on how the retrieved results enhance the generation, there are various foundational paradigms (or foundations, to put it short): they can act as an enhanced input to the generator [33], [34]; they can join as latent representations at a mid-stage of generation [35], [36]; they can contribute to the final generation results as logits [37], [38]; they can even affect or omit certain generation steps [32], [39]. Researchers have also suggested a number of improvements to strengthen the basic RAG procedure.

These techniques include targeted improvements for particular parts as well as comprehensive improvements targeted at the pipeline as a whole.

Furthermore, although the idea behind RAG first surfaced in text-to-text generation [34], this method has since found use in a wide range of fields, includeing codes [40]–[42], audios [43], [44], images [45]–[47], videos [48], [49], 3D [50], [51], knowledge [52]–[54], and artificial intelligence for science [55], [56]. Specifically, the fundamental concept and methodology of RAG are substantially uniform throughout modalities. It does, however, require some small modifications to augmentation methods, and the choice of generators and retrievers changes based on the particular modalities and applications.

The lack of a comprehensive assessment covering all foundations, advancements, and applications of RAG, in spite of the field's recent rapid expansion and expanding applications, is impeding its progress. The practical relevance of the research in this area is severely undermined by the lack of discussion on RAG foundations, which prevents RAG's full potential from being realized. Despite of query-based RAG in text-generation tasks having garnered most study interest, it is important to recognize that other RAG foundations are equally useful and have a great deal of room for expansion. Another reason is that without a broad overview of RAG applications, practitioners and academics tend to ignore RAG's advancements across a variety of modalities and are ignorant of its potential applications.. While text creation is commonly regarded as the primary use case for RAG, it is important to note that RAG development in other modalities has also started to gain traction and has produced encouraging developments. A number of modalities have a long history of being associated with retrieval procedures, which gives RAG its unique qualities. Motivated by this, our goal in this study is to offer a thorough survey that presents a methodical summary of RAG.

B. *Contribution*

This case study provides a thorough introduction to RAG, addressing its origins, improvements, uses, benchmarks, constraints, and possible future paths. We extract the fundamentals of RAG foundations, seeing applications as modifications of these principles, notwithstanding differences in retrievers and generators across modalities and workloads. The goal of this paper is to provide scholars and practitioners with recommendations and references, along with insightful information that will help advance RAG techniques and related applications. To summarize, the following is a list of contributions:

- This study performs a thorough analysis of RAG and distills the foundational abstractions of RAG for different retrievers and generators.
- Examination of the improvements made in RAG literature and outlining the strategies used to make RAG systems more efficient.
- Survey of existing AIGC methods that use RAG techniques for different modalities and tasks, showing how RAG adds value to existing generative models.
- RAG's research directions and limitations, which provide insight into possible future developments.

C. *Related Work*

Numerous surveys have appeared as RAG develops, although they only cover a portion of the subject. Specifically, they either only cover a small portion of RAG techniques for specific contexts, or they solely concentrate on one RAG foundation. Without a thorough examination of alternative modalities, the majority of the publications that are now available concentrate on text-related RAG activities that are assisted by LLMs. A fundamental review of RAG is provided in the survey by Li et al. [57], which also covers particular applications related to text production tasks. Similar to this, Asai et al.'s tutorial [58] focuses on retrieval-based language models and describes their training approaches and architectures. Meanwhile, RAG is examined in the context of LLMs in a recent survey by Gao et al. [59], with a focus on query-based RAG optimization techniques. Our approach extends RAG's reach to the full AIGC ecosystem, acknowledging its expansion outside the text domain and enabling a more thorough coverage of RAG research. Another survey, put forth by Zhao et al. [60], skips over the topic of RAG foundations and instead provides RAG applications across several modalities. Only a portion of other modalities' works are covered in another study [61]. Even though certain facets of RAG have been studied in previous research, a thorough overview including the basics, improvements, and domain-specific applicability of RAG is still lacking. The goal of this paper is to close this gap by offering an organized analysis of RAG.

II. Preliminary

A. *Overview*

The generator and the retriever are the two main modules that make up the RAG system. The generator generates the necessary contents, while the retriever looks for pertinent information in the data store. The following is how the RAG process goes: (i) The query is first sent to the retriever, which then looks for pertinent data; (ii) The original query and the retrieval results are then fed into the generator using a certain augmentation process; (iii) Lastly, the generator generates the intended results.

B. *Generator*

The era of AIGC has begun, thanks to generative AI's outstanding performance on a variety of jobs. In the RAG system, the generating module is essential. For example, transformer models are used for text-to-text tasks, VisualGPT [62] is used for image-to-text tasks, Stable Diffusion [10] is used for text-to-image tasks, Codex [2] is used for text-to-code tasks, and so on. Various generative models are used for different circumstances. Four common generators that are commonly used in RAG are introduced here: the diffusion model, GAN, LSTM, and transformer model.

1) Transformer Model: Transformer models, which combine feedforward networks, layer normalization modules, residual networks, and self-attention mechanisms, are among the highest performing models in the field of natural language processing (NLP) [63]. At each generating phase, vocabulary classification is applied to a series of latent representations obtained from tokenization and embedding to construct the final output sequence.

2) LSTM: The Recurrent Neural Network (RNN) model has a unique variant known as Long Short-Term Memory (LSTM) [64]. Cell states and gating methods are used to address the problems of exploding/vanishing gradients in long-term dependence processing. The three gates in the model—Input, Forget, and Output—filter data, while the central Cell State module stores and controls the data. It generates outputs autoregressively using the same vocabulary classification technique as transformer models.

3) Diffusion Model: A family of deep generative models known as diffusion models is capable of producing a wide range of realistic data samples, such as texts, photos, videos, molecules, and more [65]. In order to create fresh data from noise, diffusion models first add noise to the data gradually until it becomes random, then reverse the process. Neural networks and probabilistic modeling serve as the foundation for this procedure.

4) GAN: Generative Adversarial Networks (GANs) [12] are deep learning models that can generate realistic images, audio, and other data [66]. They consist of a generator and a discriminator, which compete through adversarial learning. The generator continuously improves its ability to generate realistic samples, while the discriminator continuously improves its ability to distinguish between true and false samples.

C. Retriever

Finding and obtaining pertinent information in response to an information need is known as retrieval. In particular, let's look at data sources that may be thought of as a key-value store, in which every key is associated with a value (keys and values can be the same). The goal is to use a similarity function to find the top k most similar keys to a given query in order to extract the associated values. Existing retrieval techniques can be divided into sparse retrieval, dense retrieval, and other categories based on various similarity functions. The entire process of commonly used sparse and dense retrieval may be broken down into two separate stages: (i) each object is first encoded into a particular representation, and (ii) an index is created to arrange the data source for effective search.

1) Sparse Retriever: Sparse retrieval techniques are frequently employed in document retrieval, where the documents to be searched are represented by the keys or values. This is done by making use of term matching metrics that examine word statistics from texts and create inverted indices for effective searching, such as TF-IDF [67], query probability [68], and BM25 [19]. In general, BM25 is a robust baseline for extensive online search that incorporates query token occurrences, inverse document frequency weights, and other relevant metrics. Typically, sparse retrieval uses an inverted index to arrange items in order to facilitate effective search. In specifics, every term in the query looks up a list of potential documents, which are then ranked according to their statistical rankings.

Typically, sparse retrieval uses an inverted index to arrange items in order to facilitate effective search. In specifics, every term in the query looks up a list of potential documents, which are then ranked according to their statistical rankings. Typically, sparse retrieval uses an inverted index to arrange items in order to facilitate effective search. In specifics, every term in the query looks up a list of potential documents, which are then ranked according to their statistical rankings.

2) Dense Retriever: Dense retrieval techniques, in contrast to sparse retrieval, use dense embedding vectors to represent queries and keys and create an Approximate Nearest Neighbor (ANN) index to expedite the search. This is true for every modality. Recent developments in pre-trained models (like BERT [15]) have been used to encode queries and keys separately for text data [19]. Dense Passage Retrieval (DPR) is a common term for this method. Models for encoding code data [25], audio data [69], image data [24], video data [70], and other types of data have been proposed, much like text. Typically, measures like cosine, inner product, and L2-distance are used to calculate the similarity score between dense representations.

Contrastive learning is used in dense retrieval training to make positive data more similar and negative samples less similar. To improve model quality even more, a number of hard negative techniques [71] have been put forth. ANN algorithms are used for effective searching during inference. Tree [72], [73], location sensitive hashing [74], neighbor graph indices (e.g., HNSW [75], DiskANN [76]), and combined graph and inverted indices (e.g., SPANN [22]) are some of the indices created to support ANN search.

3) Others: There are more techniques for obtaining pertinent objects besides sparse and dense retrieval [77], [78]. Some studies employ the edit distance between natural language texts [79] or abstract syntax trees (AST) of code snippets [80], [81] directly in place of computing representations. Relationships between entities in knowledge graphs act as a pre-built index for retrieval. K-hop neighbor searches can therefore be used for retrieval in RAG approaches that use knowledge graphs [82], [83]. Named Entity Recognition (NER) [84] is an additional retrieval technique in which the entities serve as keys and the query as the input.

III. METHODOLOGIES

A. RAG Foundations

1) Query-based RAG: Originating from the concept of prompt augmentation, query-based RAG easily incorporates insights from retrieved data with the user's inquiry, delivering it straight into the generator's input stage. This approach is often used in RAG applications. After being retrieved, the content is combined with the user's initial query to generate a composite input, which the generator processes to produce a response. Query-based RAG is frequently used in many different modalities.

REALM [33] uses a dual-BERT framework for text production, combining knowledge extractors with pre-trained models to expedite knowledge retrieval and integration. Lewis et al. [85] used BART as the generator to efficiently improve the generation and DPR for information retrieval. A critique module is used by SELF-RAG [86] to assess if the retrieval is necessary. Query-based RAG can be used in situations that use LLM through API calls, in addition to being interoperable with local generators. By considering the language model as a ”black box,” REPLUG [87] adheres to this paradigm and successfully incorporates pertinent external documents into the query. The top-ranked documents are reordered and integrated using a predictive reranker trained using In-Context RALM [88], which leverages BM25 for document retrieval.

The query-based paradigm has been used in a number of publications [42], [89]–[92] in the field of code to improve the efficacy of downstream tasks by incorporating contextual information from text or code into the prompt.

Recent studies on Knowledge Base Question Answering (KBQA) have also demonstrated the important benefits of integrating language and retrieval models. For example, by combining inquiries and obtained data into prompts, Uni-Parser [93], RNG-KBQA [82], and ECBRF [94] successfully increase the accuracy and performance of QA systems.

Chat-Orthopedist [95], a tool in the AI-for-Science space, uses recovered data in model prompts, facilitating in shared decision-making for teenagers with idiopathic scoliosis and increases the efficacy and accuracy of LLMs.

RetrieveGAN [45] incorporates retrieved data, such as selected picture patches and their bounding boxes, into the generator's input stage to increase the relevance and accuracy of generated images in the image generating task. Noise vectors and instance characteristics are concatenated by IC-GAN [96], which adjusts the particular conditions and details of the generated images.

RetDream [50] uses CLIP [8] to first recover pertinent 3D elements for 3D generation. During the input phase, the returned contents are combined with user input.

Frequently used in conjunction with LLM generators, query-based RAG pro- vides modular flexibility that enables the rapid integration of pretrained components for rapid deployment. Using the retrieved data in this setting requires quick design.

2) Latent Representation-based RAG: The recovered objects are used as latent representations in generative models in the latent representation-based RAG framework, thereby improving the quality of the generated information and strengthening the model's understanding capabilities.

FiD [35] and RETRO [36] are two traditional structures of latent representation- based RAG in the text field upon which numerous later works have made changes. FiD [35] combines the generated latent representations for decoding by a single decoder to generate the final output after processing each recovered paragraph, its title, and the query through separate encoders. After retrieving pertinent data for every segmented sub-query, RETRO [36] uses a brand-new module called Chunked Cross-Attention (CCA) to combine the obtained data with each sub-query token. Other significant innovative structures fall under the purview of latent representation-based RAG as well. In order to enable input chunking and, in theory, meet the long-criticized context length limits of Transformer models, a number of studies [31], [97] have integrated k Nearest Neighbor (kNN) search into transformer blocks. Kuratov et al. [98] combined Transformer with RNN, using the intermediate output of the model as the retrieval content.

FiD has become widely used in the disciplines of science and code, with applications in a variety of code-related domains [99]–[103] and AI-for-Science [55].

Several research [104]–[107] use cross-attention techniques in the visual domain to integrate their latent representations and merge retrieval outcomes. On the other hand, Li et al. [108] use an Affine Combination Module (ACM) that concatenates hidden characteristics directly between text and images.

Numerous studies [109]–[113] have used FiD and its derivatives for downstream tasks inside the knowledge domain. While TOME [114] shifts to a nuanced encoding of mentions, giving mention granularity precedence over entity representations alone, EaE [115] improves the generator's comprehension by entity-specific parameterization.

ReMoDiffuse [51] advances the field of 3D generation by introducing a semantics-modulated attention method that improves the precision of producing comparable 3D motions from textual descriptions. By combining the original diffusion process with the reference diffusion process, AMD [116] successfully converts text to 3D motion.

Koizumi et al. [43] used an LLM in the audio domain, directing the creation of audio captions by integrating encoded dense information in the attention module. Deep features are extracted from text and audio using different encoders by ReAudioLDM [117], and these characteristics are then included into the Latent Diffusion Model's (LDM) attention mechanism.

R-ConvED [48] processes retrieved video-sentence pairs using an attention mechanism and a convolutional encoder-decoder network, creating hidden states to generate captions for videos. CARE [118] integrates idea representations into a hybrid attention mechanism and presents a concept detector to generate concept probabilities. EgoInstructor [49] enhances the coherence and relevance of captions for egocentric videos by combining text and visual elements via gated-cross attention. Latent representation-based RAG combines retriever and generator hidden states and is flexible across modalities and tasks, although it necessitates extra training to align latent spaces. It makes it possible to create complex algorithms that smoothly integrate the data that has been retrieved.

3) Logit-based RAG: During the decoding phase, generative models incorporate retrieval information via logits in logit-based RAG. To calculate the probability for step-wise generation, the logits are usually merged using straightforward summation or models.

Language model probabilities and those derived from retrieval distances of identical prefixes are combined at each decoding step in the text domain by kNN-LM [37] and its version [38]. Using highly aligned tokens from a local database as output, TRIME [119] and NPM [120] are radical extensions of conventional kNNLM techniques that improve performance especially in longtail distribution circumstances.

In addition to text, logit-based RAG is also used in other modalities like code and images.

A number of research [80], [121] have also used the kNN concept in the code domain to improve final output control and attain better performance. Additionally, EDITSUM [99] incorporates prototype summaries at the logit level to enhance the quality of code summarisation. MA [122] uses the kNN-LM frame work to solve the image caption problem with positive outcomes. This makes logit-based RAG perfect for sequence creation since it uses previous data to infer current states and combines information at the logit level. It emphasises generator training and makes room for cutting-edge techniques that take advantage of probability distributions for upcoming assignments.

4) Speculative RAG: Speculative RAG looks for ways to economise resources and speed up reaction times by using retrieval rather of pure production. REST [32] allows for the creation of drafts by substituting retrieval for the tiny models used in speculative decoding [123]. GPTCache [39] creates a semantic cache to store LLM replies, hence resolving the problem of excessive latency when utilising the LLM APIs. In order to retrieve words or phrases from the documents rather than generating them, COG [124] breaks down the text generation process into a sequence of copy-and-paste operations. Cao et al. [125] suggested a novel paradigm that substitutes directly retrieved phrase level content for generation in order to remove the final result's reliance on the calibre of the first-stage retrieved content.

Sequential data is now the main use of speculative RAG. Separating the generator and the retriever makes it possible to employ pre-trained models as components directly. We can investigate a greater variety of tactics to make efficient use of the recovered content within this framework.

B. RAG Enhancements

1) Input Enhancement: The first input fed into the retriever has a significant impact on the outcome of the retrieval stage. This section presents query transformation and data augmentation as two input enhancement techniques.

Query Transformation: By altering the input query, query transformation can improve the retrieval outcome.

The original query is used by Query2doc [126] and HyDE [127] to create a faux document, which is then used as the retrieval query. Richer, pertinent information is included in the pseudo document, which aids in the retrieval of more precise results.

By using the obtained contents, TOC [128] breaks down the confusing query into several distinct sub-queries, which are then sent to the generator and combined to yield the final output.

RQ-RAG [129] deconstructs complex or ambiguous enquiries into distinct subqueries for fine-grained retrieval and combines the answers to provide a coherent response to the initial inquiry. Tayal et al. [130] improved the generator's understanding of user intent by refining the original query using context retrieval and dynamic few-shot samples.

Data Augmentation: By using methods including deleting ambiguity, updating old documents, synthesising new data, and removing extraneous information, data augmentation enhances data prior to retrieval.

Make-An-Audio [44] adds random concept audio to enhance the original audio and employs captioning and audio-text retrieval to create captions for language-free audio in order to reduce data sparsity. In order to improve model performance in response to instructional prompts, LESS [131] analyses gradient information to optimise dataset selection for downstream tasks. To pre-train the code retrieval model, ReACC [92] uses data augmentation techniques including renaming and dead code insertion. By using a ”Vocabulary for 3GPP Specifications” and matching them to user queries using a router module, TelcoRAG [132] improves the retrieval accuracy.

2) Retriever Enhancement: The information sent into the generators in RAG systems is determined by the quality of the content that is retrieved. The likelihood of model hallucinations or other deterioration rises with lower content quality. We present useful strategies to improve retrieval efficacy in this section.

Recursive Retrieval: This method involves conducting several searches to obtain more comprehensive and superior content.

ReACT [133] provides deeper information by decomposing questions for recursive retrieval using Chain-of-Thought (CoT) [134]. The best retrieval material is chosen by RATP [135] using the Monte-Carlo Tree Search for simulations. The content is then templated and sent to the generator for output. Chunk optimisation is the process of modifying chunk size to enhance retrieval outcomes.

Chunk Optimization: Chunk optimization refers to adjusting chunk size for improved retrieval results.

One of the chunk optimisation techniques used by LlamaIndex [136] is based on the ”small to big” theory. Finding finer-grained content while returning richer information is the main idea here. Sentence-window retrieval, for example, retrieves brief text passages and provides a window of pertinent sentences that encircle the recovered section. Documents are organised in a tree structure for automerge retrieval. By initially retrieving the child node, the method obtains the parent node, which contains the content of its child nodes. RAPTOR [137] uses recurrent embedding, clustering, and summarisation of text chunks until additional clustering is impractical in order to solve the lack of contextual information. This creates a multilevel tree structure. By creating a table of contents beforehand, PromptRAG [138] improves retrieval accuracy by allowing the model to choose pertinent chapters on its own based on the query. To increase recollection and produce better outcomes, Raina et al. [139] divide text fragments into smaller, more atomic assertions.

Retriever Finetuning: The core component of the RAG system, the retriever, depends on an effective embedding model [140]–[143] to feed the generator with relevant content and represent it, improving system performance.

Furthermore, domain-specific or task-related data can be used to refine embedding models with high expressive power in order to improve performance in certain domains. REPLUG [87] handles LM as black box, which updates the retriever model in response to the outcomes. Python files, api names, signatures, and descriptions are used by APICoder [89] to refine the retriever.

After retrieval, EDITSUM [99] optimises the retriever to reduce the jaccard distance between summaries. Target Similarity Tuning (TST) is used by SYNCHROMESH [81] to fine-tune the retriever after adding tree distance os ASTs to the loss. Using the same data as the generator, R-ConvED [48] optimizes the retriever. InfoNCE loss was used by Kulkarni et al. [144] to optimise the retriever.

Hybrid Retrieval: A hybrid retrieve refers to the simultaneous use of a wide range of retrieval techniques or the extraction of data from several different sources.

To increase the quality of retrieval, RAP-Gen [145], BlendedRAG [146], and ReACC [92] employ both dense and sparse retrievers. Rencos [80] retrieves similar code snippets on a syntactic level using a sparse retriever and on a semantic level using a dense retriever. BASHEXPLAINER [100] first gathers semantic data using a dense retriever, and then it gathers lexical data using a sparse retriever. RetDream [50] retrieves using text first, followed by image embedding. A retrieval evaluator in CRAG [147] determines the relevance of documents to queries and generates three retrieval replies based on confidence: a hybrid approach for unclear circumstances, Web Search if results are inaccurate, and direct use of results for Knowledge Refinement if results are accurate. By adding DKS (Dense Knowledge Similarity) and RAC (Retriever as Answer Classifier) to the retrieval phase and assessing answer relevance and knowledge applicability, Huang et al. [148] enhanced question-answering. A new type of token known as the ”acting token,” which establishes the source from which to obtain information, is introduced by UniMSRAG [149]. By combining text and drawing for fine-grained retrieval, Koley et al. [150] improve image retrieval and produce better outcomes.

Reranking: Rearranging the content that has been obtained in order to increase diversity and improve outcomes is known as the Rerank technique. In order to lessen the impact of information loss brought on by text compression into vectors, Re2G [151] uses a re-ranker [152] model after the conventional retriever. In order to eliminate redundant programs and produce a diversified set of retrieved programs, AceCoder [153] reranks the programs using a selector. Following retrieval, XRICL [154] employs an exemplar reranker based on distillation. Rangan, et al. [155] evaluate the similarity of data subsets and reranks retrieval results by using the Quantised Influence Measure, which measures statistical biases between a query and a reference. In order to create a cohesive retriever, UDAPDR [156] use multi-teacher knowledge distillation in conjunction with LLMs to economically produce synthetic queries that train domain-specific rerankers. By using a static LLM for document rating and reward model training in addition to knowledge distillation, LLM-R [157] iteratively improves its retriever. Progressive optimisation is made possible by the retriever's incremental improvement with each training cycle. Finardi et al. [158] used monoT5 as a reranker to maximise the quality of the results and incorporated reciprocal rank into the retrieval process for improved text chunk relevancy. Li et al. [159] improve the retrieval quality and factual accuracy of LLMs by incorporating a reranking module into their end-to-end RAG system.

Retrieval Tranformation: Retrieval transformation is the process of reword- ing content that has been retrieved in order to better engage the generator's potential and provide better output.

In order to simplify the generator's duty and enable precise answer prediction, FILCO [160] effectively removes unnecessary content from recovered text, separating just the relevant supporting stuff. In order to significantly reduce latency time, FiD-Light [161] first uses an encoder to transform the retrieved content into a vector, which it subsequently compresses. Using a template, RRR [162] combines the current query with the top-k documents in each round before restructuring it using LLMs that have already been trained (GPT-3.5-Turbo, etc.).

Others: There are more optimisation techniques for the retrieval process in addition to the ones mentioned above.

For instance, meta-data filtering [163] is a technique to aid in the processing of retrieved documents by filtering them for better outcomes using metadata (such as time, purpose, etc.). By asking an LLM to produce documents in response to a specific query, GENREAD [164] and GRG [165] present a revolutionary method that replaces or enhances the retrieval process. In order to improve retrieval accuracy, Multi-Head-RAG [166] uses a multi-head attention layer to capture distinct informational features and numerous embedding models to project the same text chunk into different vector spaces.

C. Generator Enhancement

The quality of the output results in RAG systems is frequently dictated by the quality of the generator. As a result, the maximum effectiveness of the entire RAG system is determined by the generator's capability.

Prompt Engineering: LLM generators in RAG systems might benefit from technologies in prompt engineering [167] that concentrate on enhancing the output quality of LLMs, such prompt compression, Stepback Prompt [168], Active Prompt [169], Chain of Thought Prompt [134], etc.

In order to speed up model inference, LLMLingua [170] uses a tiny model to condense the query's total length. This lessens the detrimental effect of extraneous information on the model and the ”Lost in the Middle” [171] issue. Using ChatGPT, ReMoDiffuse [51] breaks down intricate explanations into anatomical text scripts. To improve outcomes, ASAP [172] adds exemplar tuples—which include input code, function definitions, analysis findings, and related comments—to prompts. CEDAR [90] arranges code demonstration, question, and natural language instructions into a prompt using a pre-made prompt template. Translation pairs are added by XRICL [154] using COT technology as a transitional stage in cross-linguistic semantic parsing and inference. The Cognition Nexus method is used by ACTIVERAG [173] to calibrate LLMs' internal cognition, and COT prompt is applied when generating answers. Other modalities can be used as input by Make-An-Audio [44], which can yield far more detailed data for the process that follows.

Generator Finetuning: Among other changes, decoding tuning entails improving generator control by adjusting hyperparameters for greater variability and limiting the output vocabulary.

InferFix [91] modifies the decoder's temperature to balance the variety and calibre of returns. SYNCHROMESH [81] uses a completion engine to remove implementation flaws and restricts the decoder's output vocabulary. Finetuning the generator can improve the model's capacity to suit the retriever more accurately or have more exact domain knowledge.

RETRO combines the content of the query and retriever by fixing the retriever's parameters and using the chunked cross attention mechanism in the generator. The generator CODEGEN-MONO 350M [174] is improved by API-Coder [90] using a shuffled new file along with code blocks and API metadata. While maintaining the encoders and retriever fixed, CARE [118] trains encoders using picture, audio, and video-text pairings before optimising the decoder (generator) to concurrently decrease caption and concept identification loss. After using picture data to optimise the video generator, Animate-AStory [175] fine tunes a LoRA [176] adaptor to capture the specifics of the character's appearance. RetDream [50] uses the produced images to refine a LoRA adaptor [176].

D. Result Enhancement

In many situations, RAG results might not have the desired impact; nevertheless, there are methods for improving results that can assist mitigate this issue.

Output Rewrite: Rewriting the material produced by the generator in specific situations to satisfy the requirements of activities that come after is known as output rewrite. In order to better match the real-world code context, SARGAM [177] uses a unique Transformer in conjunction with Deletion, Placeholder, and Insertion Classifiers to enhance outputs in code-related activities. By reranking candidates according to the average of the log probabilities generated by the generator for each token, Ring [178] is able to acquire diversity outcomes. By matching the created relations with those shown in the knowledge graph's immediate neighbourhood of the query entity, CBRKBQA [54] updates the outcome.

E. RAG Pipeline Enhancement

RAG pipeline augmentation is the process of streamlining the entire RAG process to improve performance outcomes.

Adaptive Retrieval: According to certain RAG research, retrieval doesn't always improve the outcome. When the model's intrinsic parameterised information is sufficient to address pertinent concerns, over-retrieval may result in resource waste and possible misunderstanding. Thus, rule-based and model-based techniques to assessing retrieval requirement will be covered in this subsection.

Rule Based: Using probability, FLARE [179] actively determines when and whether to search during the generating process. To calculate the percentage of generation and retrieval, Efficient-KNNLM [38] includes the generation probability of KNN-LM [37] and NPM [120] along with a hyperparameter λ.

For high-level questions, Mallen et al. [27] used statistical analysis to provide accurate answers, but for low-frequency questions they used RAG. Jiang et al. [180] assessed model confidence using fit statistics, model uncertainty, and fit uncertainty to inform regression choices. In order to determine whether the deduction is appropriate, Kandpal et al. [181] investigated the relationship between the amount of relevant text and the comprehension of model knowledge.

Model-based: In order to decide whether to execute a retrieval based on the retrieve token under various user queries, Self-RAG [86] makes use of a trained generator. Ren et al. [182] employed ”Judgement Prompting” to assess LLMs' ability to respond to pertinent queries and the accuracy of their responses, which helped determine if a retrieval was required.

SKR [183] makes use of LLMs' inherent capacity to determine beforehand whether they are able to respond to the inquiry; if they do, no retrieval is necessary. In order to ascertain whether information retrieval is necessary, Rowen [184] translates a query into several languages and verifies that the responses are consistent across these languages. AdaptiveRAG [185] uses a classifier, which is a smaller LM, to dynamically determine whether to retrieve based on the query difficulty.

Iterative RAG: Instead of using a single round, iterative RAG cycles through the retrieval and creation phases again to gradually improve results.

In order to effectively utilise scattered data and enhance results, RepoCoder [186] refines queries using previously created code through an iterative retrieval-generation approach to code completion. By employing the generator's output to identify knowledge gaps, retrieve pertinent data, and inform subsequent generation cycles, ITER-RETGEN [187] iteratively improves the quality of the content. Using an iterative retrieval-augmented generator, SelfMemory [188] creates a large memory pool from which a memory selector selects an output to feed the subsequent generation cycle. RAT [189] uses a zero-shot CoT prompt to first generate material by an LLM, then retrieves information from an external knowledge store to update each thinking step.

IV. Discussion

Despute the widespread adoption of RAG, it suffers from several limitations by design.

A. Noises in Retrieval Results

Information loss in item representations and ANN search makes information retrieval fundamentally faulty. RAG systems may experience failure points due to the unavoidable noise, which may appear as irrelevant content or false information [190]. Nevertheless, current research surprisingly discovers that noisy retrieval results may improve generation quality, even while increasing retrieval accuracy seems obvious for RAG efficacy [191]. One explanation is that quick building may be facilitated by a variety of retrieval outcomes [192]. As a result, it is unclear how retrieval noise affects real applications, which causes misunderstandings regarding metric selection and retriever-generator interaction

B. Extra Overhead

In most situations, retrieval has non-negligible overhead, even if it can occasionally lower generating costs [30]–[32]. Stated differently, delay is necessarily increased by the retrieval and interaction operations. This is enhanced when RAG is used in conjunction with sophisticated enhancing techniques like iterative RAG [186] and recursive retrieval [193]. Moreover, the complexity of access and storage will rise in tandem with the size of retrieval sources [194]. The usefulness of RAG for latency-sensitive real-time systems is severely hampered by this overhead.

C. The Gap between Generators and Retrievers

The interplay between retrievers and generators necessitates careful design and optimisation because their latent spaces and goals may not coincide. Present methods either separate generation and retrieval or combine them in a middle stage. The latter could gain from combined training but hinder generality, whereas the former is more modular. Choosing an affordable engagement strategy to close the gap is difficult and requires careful consideration in real-world situations.

D. Increased System Complexity

The complexity of the system and the amount of hyper-parameters to adjust inevitably rise with the addition of retrieval. In query-based RAG, for example, a recent study discovered that employing top-k rather than a single retrieval enhances attribution but degrades fluency [195]. Other factors, such metric selection, are yet not fully investigated. Therefore, when RAG is involved, tuning the generation service calls for greater skill.

E. Lengthy Context

RAG's enormous context lengthening, especially the query-based RAG, is one of its main drawbacks, rendering it unworkable for generators with constrained context length. Furthermore, the extended context often slows down the creation process. These issues have been somewhat alleviated by research developments in long-context support [196] and quick compression [170], but at a minor cost or accuracy trade-off.

V. Conclusion

This case study discussed about an extensive and in-depth analysis of RAG in the framework of AIGC, highlighting special attention to the applications, improvements, and foundations of augmentation. We started by methodically classifying and summarising the fundamental RAG concepts, offering insights into how retrievers and generators interact. Next, we looked at the improvements made to RAG that increase its efficacy even more, whether they were made to the pipeline as a whole or to individual components. We demonstrated real-world RAG implementations in a variety of tasks and modalities to aid researchers from a wide range of fields.

References

[1] T. B. Brown, B. Mann *et al.*, "Language models are few-shot learners," in *NeurIPS*, 2020.
[2] M. Chen, J. Tworek *et al.*, "Evaluating large language models trained on code," *arXiv:2107.03374*, 2021.
[3] OpenAI, "GPT-4 technical report," *arXiv:2303.08774*, 2023.
[4] H. Touvron, T. Lavril *et al.*, "Llama: Open and efficient foundation language models," *arXiv:2302.13971*, 2023.
[5] H. Touvron, L. Martin *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv:2307.09288*, 2023.
[6] B. Rozière, J. Gehring *et al.*, "Code llama: Open foundation models for code," *arXiv:2308.12950*, 2023.
[7] A. Ramesh, M. Pavlov, G. Goh *et al.*, "Zero-shot text-to-image generation," in *ICML*, 2021.
[8] A. Ramesh, P. Dhariwal, A. Nichol *et al.*, "Hierarchical text-conditional image generation with CLIP latents," *arXiv:2204.06125*, 2022.
[9] J. Betker, G. Goh, L. Jing *et al.*, "Improving image generation with better captions," *Computer Science*, vol. 2, no. 3, p. 8, 2023.
[10] R. Rombach, A. Blattmann, D. Lorenz *et al.*, "High-resolution image synthesis with latent diffusion models," in *IEEE/CVF*, 2022.
[11] OpenAI, "Video generation models as world simulators," https://openai.com/research/video-generation-models-as-world-simulators, 2024.
[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.*, "Generative adversarial networks," *CACM*, vol. 63, no. 11, pp. 139–144, 2020.
[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[14] A. Vaswani, N. Shazeer, N. Parmar *et al.*, "Attention is all you need," in *NeurIPS*, 2017.
[15] D. Guo, S. Ren *et al.*, "Graphcodebert: Pre-training code representations with data flow," in *ICLR*, 2021.
[16] C. Raffel, N. Shazeer, A. Roberts *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *JMLR*, vol. 21, pp. 140:1–140:67, 2020.
[17] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *JMLR*, vol. 23, no. 120, pp. 1–39, 2022.
[18] J. Kaplan, S. McCandlish, T. Henighan *et al.*, "Scaling laws for neural language models," 2020.
[19] S. E. Robertson and H. Zaragoza, "The probabilistic relevance framework: BM25 and beyond," *FTIR*, vol. 3, no. 4, pp. 333–389, 2009.
[20] V. Karpukhin, B. Oguz, S. Min *et al.*, "Dense passage retrieval for open-domain question answering," in *EMNLP*, 2020.
[21] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, 2021.
[22] Q. Chen, B. Zhao, H. Wang *et al.*, "SPANN: highly-efficient billion-scale approximate nearest neighborhood search," in *NeurIPS*, 2021.
[23] R. Datta, D. Joshi, J. Li *et al.*, "Image retrieval: Ideas, influences, and trends of the new age," *CSUR*, vol. 40, no. 2, pp. 5:1–5:60, 2008.
[24] A. Radford, J. W. Kim, C. Hallacy *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.
[25] Z. Feng, D. Guo *et al.*, "Codebert: A pre-trained model for programming and natural languages," in *EMNLP Findings*, 2020.

[26] Y. Wu, K. Chen, T. Zhang *et al.*, "Large-scale contrastive language-audio pretraining with feature fusion and keyword-to-caption augmentation," in *ICASSP*, 2023.
[27] A. Mallen, A. Asai, V. Zhong *et al.*, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *ACL*, 2023.
[28] N. Carlini, F. Tramèr *et al.*, "Extracting training data from large language models," in *USENIX*, 2021.
[29] M. Kang, N. M. Gürel *et al.*, "C-RAG: certified generation risks for retrieval-augmented language models," *arXiv:2402.03181*, 2024.
[30] G. Izacard, P. Lewis, M. Lomeli *et al.*, "Atlas: Few-shot learning with retrieval augmented language models," *arXiv:2208.03299*, 2022.
[31] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy, "Memorizing transformers," in *ICLR*, 2022.
[32] Z. He, Z. Zhong, T. Cai *et al.*, "REST: retrieval-based speculative decoding," *arxiv:2311.08252*, 2023.
[33] K. Guu, K. Lee, Z. Tung *et al.*, "REALM: retrieval-augmented language model pre-training," *ICML*, 2020.
[34] P. S. H. Lewis, E. Perez, A. Piktus *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *NeurIPS*, 2020.
[35] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *EACL*, 2021.
[36] S. Borgeaud, A. Mensch *et al.*, "Improving language models by retrieving from trillions of tokens," in *ICML*, 2022.
[37] U. Khandelwal, O. Levy, D. Jurafsky *et al.*, "Generalization through memorization: Nearest neighbor language models," in *ICLR*, 2020.
[38] J. He, G. Neubig, and T. Berg-Kirkpatrick, "Efficient nearest neighbor language models," in *EMNLP*, 2021.
[39] zilliztech. (2023) Gptcache. [Online]. Available: https://github.com/zilliztech/GPTCache
[40] M. R. Parvez, W. U. Ahmad *et al.*, "Retrieval augmented code generation and summarization," in *EMNLP Findings*, 2021.
[41] W. U. Ahmad, S. Chakraborty, B. Ray *et al.*, "Unified pre-training for program understanding and generation," in *NAACL-HLT*, 2021.
[42] S. Zhou, U. Alon, F. F. Xu *et al.*, "Docprompting: Generating code by retrieving the docs," in *ICLR*, 2023.
[43] Y. Koizumi, Y. Ohishi *et al.*, "Audio captioning using pre-trained large-scale language model guided by audio-based similar caption retrieval," *arXiv:2012.07331*, 2020.
[44] R. Huang, J. Huang, D. Yang *et al.*, "Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models," in *ICML*, 2023.
[45] H.-Y. Tseng, H.-Y. Lee *et al.*, "Retrievegan: Image synthesis via differentiable patch retrieval," in *ECCV*, 2020.
[46] S. Sarto, M. Cornia, L. Baraldi, and R. Cucchiara, "Retrieval-augmented transformer for image captioning," in *CBMI*, 2022.
[47] R. Ramos, B. Martins *et al.*, "Smallcap: lightweight image captioning prompted with retrieval augmentation," in *CVPR*, 2023.
[48] J. Chen, Y. Pan, Y. Li *et al.*, "Retrieval augmented convolutional encoder-decoder networks for video captioning," *TOMCCAP*, vol. 19, no. 1s, pp. 48:1–48:24, 2023.
[49] J. Xu, Y. Huang, J. Hou *et al.*, "Retrieval-augmented egocentric video captioning," *arXiv:2401.00789*, 2024.
[50] J. Seo, S. Hong *et al.*, "Retrieval-augmented score distillation for text-to-3d generation," *arXiv:2402.02972*, 2024.
[51] M. Zhang, X. Guo, L. Pan *et al.*, "Remodiffuse: Retrieval-augmented motion diffusion model," in *ICCV*, 2023.
[52] X. Hu, X. Wu, Y. Shu, and Y. Qu, "Logical form generation via multi-task learning for complex question answering over knowledge bases," in *COLING*, 2022.
[53] X. Huang, J. Kim, and B. Zou, "Unseen entity handling in complex question answering over knowledge base via language generation," in *EMNLP Findings*, 2021.
[54] R. Das, M. Zaheer, D. Thai *et al.*, "Case-based reasoning for natural language queries over knowledge bases," in *EMNLP*, 2021.
[55] Z. Wang, W. Nie, Z. Qiao *et al.*, "Retrieval-based controllable molecule generation," in *ICLR*, 2022.
[56] Q. Jin, Y. Yang, Q. Chen, and Z. Lu, "Genegpt: Augmenting large language models with domain tools for improved access to biomedical information," *Bioinformatics*, vol. 40, no. 2, p. btae075, 2024.
[57] H. Li, Y. Su, D. Cai *et al.*, "A survey on retrieval-augmented text generation," *arxiv:2202.01110*, 2022.
[58] A. Asai, S. Min, Z. Zhong, and D. Chen, "Acl 2023 tutorial: Retrieval-based language models and applications," *ACL 2023*, 2023.
[59] Y. Gao, Y. Xiong *et al.*, "Retrieval-augmented generation for large language models: A survey," *arxiv:2312.10997*, 2023.
[60] R. Zhao, H. Chen *et al.*, "Retrieving multimodal information for augmented generation: A survey," in *EMNLP*, 2023.
[61] Y. Ding, W. Fan *et al.*, "A survey on rag meets llms: Towards retrieval-augmented large language models," *arXiv:2405.06211*, 2024.
[62] J. Chen, H. Guo, K. Yi *et al.*, "Visualgpt: Data-efficient adaptation of pretrained language models for image captioning," in *CVPR*, 2022.
[63] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *CSUR*, vol. 55, no. 6, pp. 109:1–109:28, 2023.
[64] G. V. Houdt *et al.*, "A review on the long short-term memory model," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5929–5955, 2020.
[65] L. Yang, Z. Zhang *et al.*, "Diffusion models: A comprehensive survey of methods and applications," *CSUR*, vol. 56, no. 4, pp. 1–39, 2023.
[66] J. Gui, Z. Sun, Y. Wen *et al.*, "A review on generative adversarial networks: Algorithms, theory, and applications," *TKDE*, vol. 35, no. 4, pp. 3313–3332, 2023.
[67] S. E. Robertson and S. Walker, "On relevance weights with little relevance information," in *SIGIR*, 1997.
[68] J. D. Lafferty and C. Zhai, "Document language models, query models, and risk minimization for information retrieval," in *SIGIR*, 2001.
[69] S. Hershey, S. Chaudhuri *et al.*, "CNN architectures for large-scale audio classification," in *ICASSP*, 2017.
[70] J. Dong, X. Li, C. Xu *et al.*, "Dual encoding for zero-example video retrieval," in *CVPR*, 2019.
[71] L. Xiong, C. Xiong, Y. Li *et al.*, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *ICLR*, 2021.
[72] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *CACM*, vol. 18, no. 9, pp. 509–517, 1975.
[73] W. Li, C. Feng, D. Lian *et al.*, "Learning balanced tree indexes for large-scale vector retrieval," in *SIGKDDg*, 2023.
[74] M. Datar, N. Immorlica, P. Indyk *et al.*, "Locality-sensitive hashing scheme based on p-stable distributions," in *SCG*, 2004.
[75] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *TPAMI*, vol. 42, no. 4, pp. 824–836, 2018.
[76] S. Jayaram Subramanya, F. Devvrit *et al.*, "Diskann: Fast accurate billion-point nearest neighbor search on a single node," *NeurIPS*, 2019.
[77] Y. Wang, Y. Hou, H. Wang *et al.*, "A neural corpus indexer for document retrieval," in *NeurIPS*, 2022.
[78] H. Zhang, Y. Wang, Q. Chen *et al.*, "Model-enhanced vector index," in *NeurIPS*, 2023.
[79] S. A. Hayati, R. Olivier, P. Avvaru *et al.*, "Retrieval-based neural code generation," in *EMNLP*, 2018.
[80] J. Zhang, X. Wang, H. Zhang *et al.*, "Retrieval-based neural source code summarization," in *ICSE*, 2020.
[81] G. Poesia, A. Polozov, V. Le *et al.*, "Synchromesh: Reliable code generation from pre-trained language models," in *ICLR*, 2022.
[82] X. Ye, S. Yavuz *et al.*, "RNG-KBQA: generation augmented iterative ranking for knowledge base question answering," in *ACL*, 2022.
[83] Y. Shu *et al.*, "TIARA: multi-grained retrieval for robust question answering over large knowledge bases," *arXiv:2210.12925*, 2022.
[84] X. V. Lin, R. Socher *et al.*, "Bridging textual and tabular data for cross-domain text-to-sql semantic parsing," *arXiv:2012.12627*, 2020.
[85] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.
[86] A. Asai, Z. Wu, Y. Wang *et al.*, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," *arxiv:2310.11511*, 2023.
[87] W. Shi, S. Min, M. Yasunaga *et al.*, "Replug: Retrieval-augmented black-box language models," *arXiv:2301.12652*, 2023.
[88] O. Ram, Y. Levine, I. Dalmedigos *et al.*, "In-context retrieval-augmented language models," *arXiv:2302.00083*, 2023.
[89] D. Zan, B. Chen, Z. Lin *et al.*, "When language model meets private library," in *EMNLP Findings*, 2022.
[90] N. Nashid, M. Sintaha, and A. Mesbah, "Retrieval-based prompt selection for code-related few-shot learning," in *ICSE*, 2023.
[91] M. Jin, S. Shahriar, M. Tufano *et al.*, "Inferfix: End-to-end program repair with llms," in *ESEC/FSE*, 2023.
[92] S. Lu, N. Duan, H. Han *et al.*, "Reacc: A retrieval-augmented code completion framework," in *ACL*, 2022.

[93] Y. Liu, S. Yavuz, R. Meng, D. Radev, C. Xiong, and Y. Zhou, "Uni-parser: Unified semantic parser for question answering on knowledge base and database," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 8858–8869. [Online]. Available: https://aclanthology.org/2022.emnlp-main.605/
[94] Z. Yang, X. Du, E. Cambria *et al.*, "End-to-end case-based reasoning for commonsense knowledge base completion," in *EACL*, 2023.
[95] W. Shi, Y. Zhuang, Y. Zhu *et al.*, "Retrieval-augmented large language models for adolescent idiopathic scoliosis patients in shared decision-making," in *ACM-BCB*, 2023.
[96] A. Casanova, M. Careil, J. Verbeek *et al.*, "Instance-conditioned gan," in *NeurIPS*, 2021.
[97] A. Bertsch, U. Alon, G. Neubig, and M. R. Gormley, "Unlimiformer: Long-range transformers with unlimited length input," 2024.
[98] Y. Kuratov, A. Bulatov *et al.*, "In search of needles in a 10m haystack: Recurrent memory finds what llms miss," *arXiv:2402.10790*, 2024.
[99] J. Li, Y. Li, G. Li *et al.*, "Editsum: A retrieve-and-edit framework for source code summarization," in *ASE*, 2021.
[100] C. Yu, G. Yang, X. Chen *et al.*, "Bashexplainer: Retrieval-augmented bash code comment generation based on fine-tuned codebert," in *ICSME*, 2022.
[101] T. B. Hashimoto, K. Guu, Y. Oren, and P. Liang, "A retrieve-and-edit framework for predicting structured outputs," in *NeurIPS*, 2018.
[102] B. Wei, Y. Li, G. Li *et al.*, "Retrieve and refine: Exemplar-based neural comment generation," in *ASE*, 2020.
[103] E. Shi, Y. Wang, W. Tao *et al.*, "RACE: retrieval-augmented commit message generation," in *EMNLP*, 2022.
[104] W. Chen, H. Hu, C. Saharia, and W. W. Cohen, "Re-imagen: Retrieval-augmented text-to-image generator," in *ICLR*, 2023.
[105] S. Sheynin, O. Ashual, A. Polyak *et al.*, "Knn-diffusion: Image generation via large-scale retrieval," in *ICLR*, 2023.
[106] A. Blattmann, R. Rombach, K. Oktay *et al.*, "Retrieval-augmented diffusion models," in *NeurIPS*, 2022.
[107] R. Rombach, A. Blattmann, and B. Ommer, "Text-guided synthesis of artistic images with retrieval-augmented diffusion models," *arXiv:2207.13038*, 2022.
[108] B. Li, P. H. Torr, and T. Lukasiewicz, "Memory-driven text-to-image generation," *arXiv:2208.07022*, 2022.
[109] B. Oguz, X. Chen, V. Karpukhin *et al.*, "Unik-qa: Unified representations of structured and unstructured knowledge for open-domain question answering," in *NAACL Findings*, 2022.
[110] D. Yu, S. Zhang *et al.*, "Decaf: Joint decoding of answers and logical forms for question answering over knowledge bases," in *ICLR*, 2023.
[111] G. Dong, R. Li, S. Wang *et al.*, "Bridging the kb-text gap: Leveraging structured knowledge-aware pre-training for KBQA," in *CIKM*, 2023.
[112] K. Wang, F. Duan, S. Wang *et al.*, "Knowledge-driven cot: Exploring faithful reasoning in llms for knowledge-intensive question answering," *arXiv:2308.13259*, 2023.
[113] D. Yu and Y. Yang, "Retrieval-enhanced generative model for large-scale knowledge graph completion," in *SIGIR*, 2023.
[114] M. de Jong, Y. Zemlyanskiy, N. FitzGerald *et al.*, "Mention memory: incorporating textual knowledge into transformers through entity mention attention," in *ICLR*, 2021.
[115] T. Févry, L. B. Soares *et al.*, "Entities as experts: Sparse memory access with entity supervision," in *EMNLP*, 2020.
[116] B. Jing, Y. Zhang, Z. Song *et al.*, "Amd: Anatomical motion diffusion with interpretable motion decomposition and fusion," in *AAAI*, 2024.
[117] Y. Yuan, H. Liu, X. Liu *et al.*, "Retrieval-augmented text-to-audio generation," in *ICASSP*, 2024.
[118] B. Yang, M. Cao, and Y. Zou, "Concept-aware video captioning: Describing videos with effective prior information," *TIP*, vol. 32, pp. 5366–5378, 2023.
[119] Z. Zhong, T. Lei, and D. Chen, "Training language models with memory augmentation," in *EMNLP*, 2022.
[120] S. Min, W. Shi, M. Lewis *et al.*, "Nonparametric masked language modeling," in *ACL Findings*, 2023.
[121] X. Zhang, Y. Zhou, G. Yang, and T. Chen, "Syntax-aware retrieval augmented code generation," in *EMNLP Findings*, 2023.
[122] Z. Fei, "Memory-augmented image captioning," in *AAAI*, 2021.
[123] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *ICML*, 2023.
[124] T. Lan, D. Cai, Y. Wang *et al.*, "Copy is all you need," in *ICLR*, 2023.
[125] B. Cao, D. Cai, L. Cui *et al.*, "Retrieval is accurate generation," *arXiv:2402.17532*, 2024.
[126] L. Wang, N. Yang, and F. Wei, "Query2doc: Query expansion with large language models," in *EMNLP*, 2023.
[127] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," in *ACL*, 2023.
[128] G. Kim, S. Kim, B. Jeon *et al.*, "Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models," in *EMNLP*, 2023.
[129] C.-M. Chan, C. Xu *et al.*, "Rq-rag: Learning to refine queries for retrieval augmented generation," *arXiv:2404.00610*, 2024.
[130] A. Tayal and A. Tyagi, "Dynamic contexts for generating suggestion questions in rag based conversational systems," in *WWW'24 Companion*, 2024.
[131] M. Xia, S. Malladi, S. Gururangan *et al.*, "LESS: selecting influential data for targeted instruction tuning," *arXiv:2402.04333*, 2024.
[132] A.-L. Bornea, F. Ayed *et al.*, "Telco-rag: Navigating the challenges of retrieval-augmented language models for telecommunications," *arXiv:2404.15939*, 2024.
[133] S. Yao, J. Zhao, D. Yu *et al.*, "React: Synergizing reasoning and acting in language models," in *ICLR*, 2023.
[134] J. Wei, X. Wang, D. Schuurmans *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," in *NeurIPS*, 2022.
[135] T. Pouplin, H. Sun, S. Holt, and M. Van der Schaar, "Retrieval-augmented thought process as sequential decision making," *arXiv:2402.07812*, 2024.
[136] J. Liu, "LlamaIndex," 11 2022. [Online]. Available: https://github.com/jerryjliu/llama_index
[137] P. Sarthi, S. Abdullah, A. Tuli *et al.*, "Raptor: Recursive abstractive processing for tree-organized retrieval," in *ICLR*, 2023.
[138] B. Kang, J. Kim *et al.*, "Prompt-rag: Pioneering vector embedding-free retrieval-augmented generation in niche domains, exemplified by korean medicine," *arXiv:2401.11246*, 2024.
[139] V. Raina *et al.*, "Question-based retrieval using atomic units for enterprise rag," *arXiv:2405.12363*, 2024.
[140] S. Xiao, Z. Liu, P. Zhang *et al.*, "C-pack: Packaged resources to advance general chinese embedding," *arxiv:2309.07597*, 2023.
[141] J. Chen, S. Xiao, P. Zhang *et al.*, "Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *arxiv:2309.07597*, 2023.
[142] S. Xiao, Z. Liu, P. Zhang, and X. Xing, "Lm-cocktail: Resilient tuning of language models via model merging," *arxiv:2311.13534*, 2023.
[143] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J.-Y. Nie, "Retrieve anything to augment large language models," *arxiv:2310.07554*, 2023.
[144] M. Kulkarni, P. Tangarajan, K. Kim *et al.*, "Reinforcement learning for optimizing RAG for domain chatbots," *arXiv:2401.06800*, 2024.
[145] W. Wang, Y. Wang *et al.*, "Rap-gen: Retrieval-augmented patch generation with codet5 for automatic program repair," in *ESEC/FSE*, 2023.
[146] K. Sawarkar, A. Mangal *et al.*, "Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers," *arXiv:2404.07220*, 2024.
[147] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective retrieval augmented generation," *arXiv:2401.15884*, 2024.
[148] W. Huang, M. Lapata, P. Vougiouklis *et al.*, "Retrieval augmented generation with rich answer encoding," in *IJCNLP-AACL*, 2023.
[149] H. Wang, W. Huang, Y. Deng *et al.*, "Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems," *arXiv:2401.13256*, 2024.
[150] S. Koley, A. K. Bhunia *et al.*, "You'll never walk alone: A sketch and text duet for fine-grained image retrieval," in *CVPR*, 2024.
[151] M. R. Glass, G. Rossiello, M. F. M. Chowdhury *et al.*, "Re2g: Retrieve, rerank, generate," in *NAACL*, 2022.
[152] R. F. Nogueira and K. Cho, "Passage re-ranking with BERT," *arxiv:1901.04085*, 2019.
[153] J. Li, Y. Zhao, Y. Li *et al.*, "Acecoder: Utilizing existing code to enhance code generation," *arXiv:2303.17780*, 2023.
[154] P. Shi, R. Zhang, H. Bai, and J. Lin, "XRICL: cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-sql semantic parsing," in *EMNLP Findings*, 2022.
[155] K. Rangan and Y. Yin, "A fine-tuning enhanced rag system with quantized influence measure as ai judge," *arXiv:2402.17081*, 2024.
[156] J. Saad-Falcon, O. Khattab, K. Santhanam *et al.*, "Udapdr: Unsupervised domain adaptation via llm prompting and distillation of rerankers," in *EMNLP*, 2023.

[157] L. Wang, N. Yang, and F. Wei, "Learning to retrieve in-context examples for large language models," *arXiv:2307.07164*, 2023.

[158] P. Finardi, L. Avila *et al.*, "The chronicles of rag: The retriever, the chunk and the generator," *arXiv:2401.07883*, 2024.

[159] J. Li, Y. Yuan, and Z. Zhang, "Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases," *arXiv:2403.10446*, 2024.

[160] Z. Wang, J. Araki, Z. Jiang *et al.*, "Learning to filter context for retrieval-augmented generation," *arxiv:2311.08377*, 2023.

[161] S. Hofstätter, J. Chen, K. Raman, and H. Zamani, "Fid-light: Efficient and effective retrieval-augmented text generation," in *SIGIR*, 2023.

[162] D. Arora, A. Kini, S. R. Chowdhury *et al.*, "Gar-meets-rag paradigm for zero-shot information retrieval," *arXiv:2310.20158*, 2023.

[163] https://www.pinecone.io.

[164] W. Yu, D. Iter *et al.*, "Generate rather than retrieve: Large language models are strong context generators," *arXiv:2209.10063*, 2022.

[165] A. Abdallah and A. Jatowt, "Generator-retriever-generator: A novel approach to open-domain question answering," *arXiv:2307.11278*, 2023.

[166] M. Besta, A. Kubicek *et al.*, "Multi-head rag: Solving multi-aspect problems with llms," *arXiv:2406.05085*, 2024.

[167] E. Saravia, "Prompt Engineering Guide," *https://github.com/dair-ai/Prompt-Engineering-Guide*, 12 2022.

[168] H. S. Zheng, S. Mishra *et al.*, "Take a step back: Evoking reasoning via abstraction in large language models," *arxiv:2310.06117*, 2023.

[169] S. Diao, P. Wang, Y. Lin, and T. Zhang, "Active prompting with chain-of-thought for large language models," *arxiv:2302.12246*, 2023.

[170] H. Jiang, Q. Wu, C. Lin *et al.*, "Llmlingua: Compressing prompts for accelerated inference of large language models," in *EMNLP*, 2023.

[171] N. F. Liu, K. Lin, J. Hewitt *et al.*, "Lost in the middle: How language models use long contexts," *arxiv:2307.03172*, 2023.

[172] T. Ahmed, K. S. Pai, P. Devanbu, and E. T. Barr, "Automatic semantic augmentation of language model prompts (for code summarization)," *arXiv:2304.06815*, 2024.

[173] Z. Xu, Z. Liu, Y. Liu *et al.*, "Activerag: Revealing the treasures of knowledge via active learning," *arXiv:2402.13547*, 2024.

[174] E. Nijkamp, B. Pang, H. Hayashi *et al.*, "A conversational paradigm for program synthesis," *arxiv:2203.13474*, 2022.

[175] Y. He, M. Xia, H. Chen *et al.*, "Animate-a-story: Storytelling with retrieval-augmented video generation," *arXiv:2307.06940*, 2023.

[176] E. J. Hu, Y. Shen, P. Wallis *et al.*, "Lora: Low-rank adaptation of large language models," in *ICLR*, 2022.

[177] C. Liu, P. Çetin, Y. Patodia *et al.*, "Automated code editing with search-generate-modify," *arXiv:2306.06490*, 2023.

[178] H. Joshi, J. P. C. Sánchez, S. Gulwani *et al.*, "Repair is nearly generation: Multilingual program repair with llms," in *AAAI*, 2023.

[179] Z. Jiang, F. F. Xu, L. Gao *et al.*, "Active retrieval augmented generation," *arXiv:2305.06983*, 2023.

[180] Z. Jiang, J. Araki, H. Ding, and G. Neubig, "How can we know *When* language models know? on the calibration of language models for question answering," *TACL*, 2021.

[181] N. Kandpal, H. Deng, A. Roberts *et al.*, "Large language models struggle to learn long-tail knowledge," in *ICML*, 2023.

[182] R. Ren, Y. Wang, Y. Qu *et al.*, "Investigating the factual knowledge boundary of large language models with retrieval augmentation," *arxiv:2307.11019*, 2023.

[183] Y. Wang, P. Li, M. Sun, and Y. Liu, "Self-knowledge guided retrieval augmentation for large language models," in *EMNLP Findings*, 2023.

[184] H. Ding, L. Pang, Z. Wei *et al.*, "Retrieve only when it needs: Adaptive retrieval augmentation for hallucination mitigation in large language models," *arXiv:2402.10612*, 2024.

[185] S. Jeong, J. Baek, S. Cho *et al.*, "Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity," *arXiv:2403.14403*, 2024.

[186] F. Zhang, B. Chen *et al.*, "Repocoder: Repository-level code completion through iterative retrieval and generation," in *EMNLP*, 2023.

[187] Z. Shao, Y. Gong, Y. Shen *et al.*, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," in *EMNLP Findings*, 2023.

[188] X. Cheng, D. Luo, X. Chen *et al.*, "Lift yourself up: Retrieval-augmented text generation with self-memory," in *NeurIPS*, 2023.

[189] Z. Wang, A. Liu, H. Lin *et al.*, "Rat: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation," *arXiv:2403.05313*, 2024.

[190] S. Barnett, S. Kurniawan, S. Thudumu *et al.*, "Seven failure points when engineering a retrieval augmented generation system," *arXiv:2401.05856*, 2024.

[191] F. Cuconasu, G. Trappolini, F. Siciliano *et al.*, "The power of noise: Redefining retrieval for RAG systems," *arXiv:2401.14887*, 2024.

[192] L. Qiu, P. Shaw, P. Pasupat *et al.*, "Evaluating the impact of model scale for compositional generalization in semantic parsing," *arXiv:2205.12253*, 2022.

[193] R. Jagerman, H. Zhuang, Z. Qin *et al.*, "Query expansion by prompting large language models," *arxiv:2305.03653*, 2023.

[194] H. Zhang, P. Zhao, X. Miao *et al.*, "Experimental analysis of large-scale learnable vector storage compression," *VLDB*, 2023.

[195] R. Aksitov, C. Chang, D. Reitter *et al.*, "Characterizing attribution and fluency tradeoffs for retrieval-augmented large language models," *arXiv:2302.05578*, 2023.

[196] C. Han, Q. Wang, W. Xiong *et al.*, "Lm-infinite: Simple on-the-fly length generalization for large language models," *arXiv:2308.16137*, 2023.