Retrieval-Augmented Generation for Large Language Models: A Survey

Yunfan Gao[a], Yun Xiong[b], Xinyu Gao[b], Kangxiang Jia[b], Jinliu Pan[b], Yuxi Bi[c], Yi Dai[a], Jiawei Sun[a], Meng Wang[c], and Haofen Wang [a,c]

[a]Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University
[b]Shanghai Key Laboratory of Data Science, School of Computer Science, Fudan University
[c]College of Design and Innovation, Tongji University

***Abstract*—Large Language Models (LLMs) showcase impressive capabilities but encounter challenges like hallucination, outdated knowledge, and non-transparent, untraceable reasoning processes. Retrieval-Augmented Generation (RAG) has emerged as a promising solution by incorporating knowledge from external databases. This enhances the accuracy and credibility of the generation, particularly for knowledge-intensive tasks, and allows for continuous knowledge updates and integration of domain-specific information. RAG synergistically merges LLMs' intrinsic knowledge with the vast, dynamic repositories of external databases. This comprehensive review paper offers a detailed examination of the progression of RAG paradigms, encompassing the Naive RAG, the Advanced RAG, and the Modular RAG. It meticulously scrutinizes the tripartite foundation of RAG frameworks, which includes the retrieval, the generation and the augmentation techniques. The paper highlights the state-of-the-art technologies embedded in each of these critical components, providing a profound understanding of the advancements in RAG systems. Furthermore, this paper introduces up-to-date evaluation framework and benchmark. At the end, this article delineates the challenges currently faced and points out prospective avenues for research and development [1].**

***Index Terms*—Large language model, retrieval-augmented generation, natural language processing, information retrieval**

I. Introduction

LARGE language models (LLMs) have achieved remarkable success, though they still face significant limitations, especially in domain-specific or knowledge-intensive tasks [1], notably producing "hallucinations" [2] when handling queries beyond their training data or requiring current information. To overcome challenges, Retrieval-Augmented Generation (RAG) enhances LLMs by retrieving relevant document chunks from external knowledge base through semantic similarity calculation. By referencing external knowledge, RAG effectively reduces the problem of generating factually incorrect content. Its integration into LLMs has resulted in widespread adoption, establishing RAG as a key technology in advancing chatbots and enhancing the suitability of LLMs for real-world applications.

RAG technology has rapidly developed in recent years, and the technology tree summarizing related research is shown in Figure 1. The development trajectory of RAG in the era of large models exhibits several distinct stage characteristics. Initially, RAG's inception coincided with the rise of the Transformer architecture, focusing on enhancing language models by incorporating additional knowledge through Pre-Training Models (PTM). This early stage was characterized by foundational work aimed at refining pre-training techniques [3]–[5].The subsequent arrival of ChatGPT [6] marked a pivotal moment, with LLM demonstrating powerful in context learning (ICL) capabilities. RAG research shifted towards providing better information for LLMs to answer more complex and knowledge-intensive tasks during the inference stage, leading to rapid development in RAG studies. As research progressed, the enhancement of RAG was no longer limited to the inference stage but began to incorporate more with LLM fine-tuning techniques.

The burgeoning field of RAG has experienced swift growth, yet it has not been accompanied by a systematic synthesis that could clarify its broader trajectory. This survey endeavors to fill this gap by mapping out the RAG process and charting its evolution and anticipated future paths, with a focus on the integration of RAG within LLMs. This paper considers both technical paradigms and research methods, summarizing three main research paradigms from over 100 RAG studies, and analyzing key technologies in the core stages of "Retrieval," "Generation," and "Augmentation." On the other hand, current research tends to focus more on methods, lacking analysis and summarization of how to evaluate RAG. This paper comprehensively reviews the downstream tasks, datasets, benchmarks, and evaluation methods applicable to RAG. Overall, this paper sets out to meticulously compile and categorize the foundational technical concepts, historical progression, and the spectrum of RAG methodologies and applications that have emerged post-LLMs. It is designed to equip readers and professionals with a detailed and structured understanding of both large models and RAG. It aims to illuminate the evolution of retrieval augmentation techniques, assess the strengths and weaknesses of various approaches in their respective contexts, and speculate on upcoming trends and innovations.

Our contributions are as follows:

- In this survey, we present a thorough and systematic review of the state-of-the-art RAG methods, delineating its evolution through paradigms including naive RAG,

Corresponding Author.Email:haofen.wang@tongji.edu.cn

[1]Resources are available at https://github.com/Tongji-KGLLM/RAG-Survey

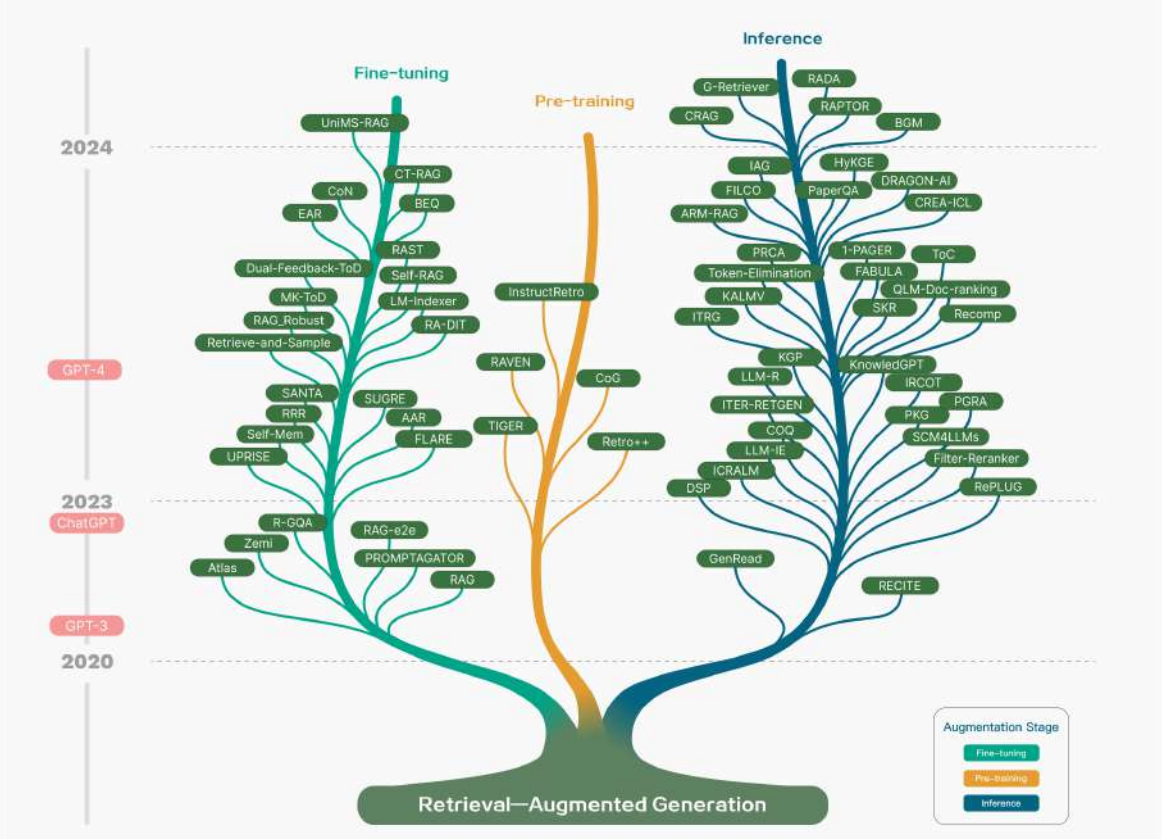


Fig. 1. Technology tree of RAG research. The stages of involving RAG mainly include pre-training, fine-tuning, and inference. With the emergence of LLMs, research on RAG initially focused on leveraging the powerful in context learning abilities of LLMs, primarily concentrating on the inference stage. Subsequent research has delved deeper, gradually integrating more with the fine-tuning of LLMs. Researchers have also been exploring ways to enhance language models in the pre-training stage through retrieval-augmented techniques.

advanced RAG, and modular RAG. This review contextualizes the broader scope of RAG research within the landscape of LLMs.

- We identify and discuss the central technologies integral to the RAG process, specifically focusing on the aspects of "Retrieval", "Generation" and "Augmentation", and delve into their synergies, elucidating how these components intricately collaborate to form a cohesive and effective RAG framework.
- We have summarized the current assessment methods of RAG, covering 26 tasks, nearly 50 datasets, outlining the evaluation objectives and metrics, as well as the current evaluation benchmarks and tools. Additionally, we anticipate future directions for RAG, emphasizing potential enhancements to tackle current challenges.

The paper unfolds as follows: Section II introduces the main concept and current paradigms of RAG. The following three sections explore core components—"Retrieval", "Generation" and "Augmentation", respectively. Section III focuses on optimization methods in retrieval,including indexing, query and embedding optimization. Section IV concentrates on post-retrieval process and LLM fine-tuning in generation. Section V analyzes the three augmentation processes. Section VI focuses on RAG's downstream tasks and evaluation system. Section VII mainly discusses the challenges that RAG currently faces and its future development directions. At last, the paper concludes in Section VIII.

II. Overview of RAG

A typical application of RAG is illustrated in Figure 2. Here, a user poses a question to ChatGPT about a recent, widely discussed news. Given ChatGPT's reliance on pre-training data, it initially lacks the capacity to provide updates on recent developments. RAG bridges this information gap by sourcing and incorporating knowledge from external databases. In this case, it gathers relevant news articles related to the user's query. These articles, combined with the original question, form a comprehensive prompt that empowers LLMs to generate a well-informed answer.

The RAG research paradigm is continuously evolving, and we categorize it into three stages: Naive RAG, Advanced RAG, and Modular RAG, as showed in Figure 3. Despite RAG method are cost-effective and surpass the performance of the native LLM, they also exhibit several limitations. The development of Advanced RAG and Modular RAG is a response to these specific shortcomings in Naive RAG.

A. Naive RAG

The Naive RAG research paradigm represents the earliest methodology, which gained prominence shortly after the

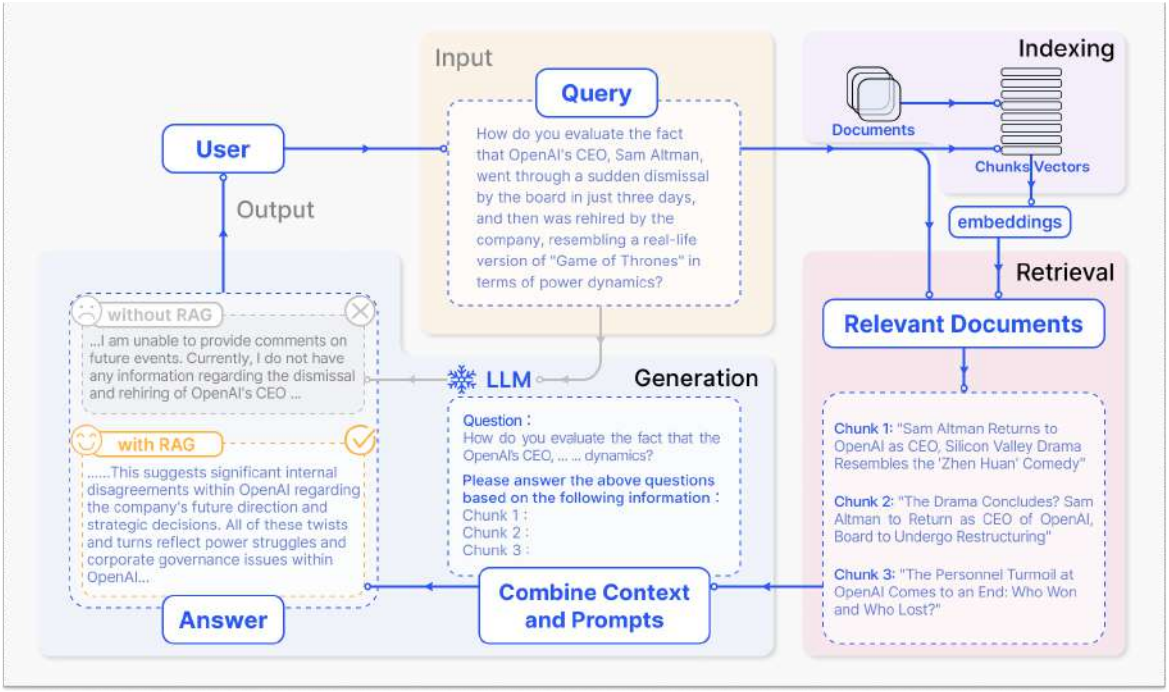


Fig. 2. A representative instance of the RAG process applied to question answering. It mainly consists of 3 steps. 1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database. 2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity. 3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer.

widespread adoption of ChatGPT. The Naive RAG follows a traditional process that includes indexing, retrieval, and generation, which is also characterized as a "Retrieve-Read" framework [7].

Indexing starts with the cleaning and extraction of raw data in diverse formats like PDF, HTML, Word, and Markdown, which is then converted into a uniform plain text format. To accommodate the context limitations of language models, text is segmented into smaller, digestible chunks. Chunks are then encoded into vector representations using an embedding model and stored in vector database. This step is crucial for enabling efficient similarity searches in the subsequent retrieval phase.

Retrieval. Upon receipt of a user query, the RAG system employs the same encoding model utilized during the indexing phase to transform the query into a vector representation. It then computes the similarity scores between the query vector and the vector of chunks within the indexed corpus. The system prioritizes and retrieves the top K chunks that demonstrate the greatest similarity to the query. These chunks are subsequently used as the expanded context in prompt.

Generation. The posed query and selected documents are synthesized into a coherent prompt to which a large language model is tasked with formulating a response. The model's approach to answering may vary depending on task-specific criteria, allowing it to either draw upon its inherent parametric knowledge or restrict its responses to the information contained within the provided documents. In cases of ongoing dialogues, any existing conversational history can be integrated into the prompt, enabling the model to engage in multi-turn dialogue interactions effectively.

However, Naive RAG encounters notable drawbacks:

Retrieval Challenges. The retrieval phase often struggles with precision and recall, leading to the selection of misaligned or irrelevant chunks, and the missing of crucial information.

Generation Difficulties. In generating responses, the model may face the issue of hallucination, where it produces content not supported by the retrieved context. This phase can also suffer from irrelevance, toxicity, or bias in the outputs, detracting from the quality and reliability of the responses.

Augmentation Hurdles. Integrating retrieved information with the different task can be challenging, sometimes resulting in disjointed or incoherent outputs. The process may also encounter redundancy when similar information is retrieved from multiple sources, leading to repetitive responses. Determining the significance and relevance of various passages and ensuring stylistic and tonal consistency add further complexity. Facing complex issues, a single retrieval based on the original query may not suffice to acquire adequate context information.

Moreover, there's a concern that generation models might overly rely on augmented information, leading to outputs that simply echo retrieved content without adding insightful or synthesized information.

B. Advanced RAG

Advanced RAG introduces specific improvements to overcome the limitations of Naive RAG. Focusing on enhancing retrieval quality, it employs pre-retrieval and post-retrieval strategies. To tackle the indexing issues, Advanced RAG refines its indexing techniques through the use of a sliding window approach, fine-grained segmentation, and the incorporation of metadata. Additionally, it incorporates several optimization methods to streamline the retrieval process [8].

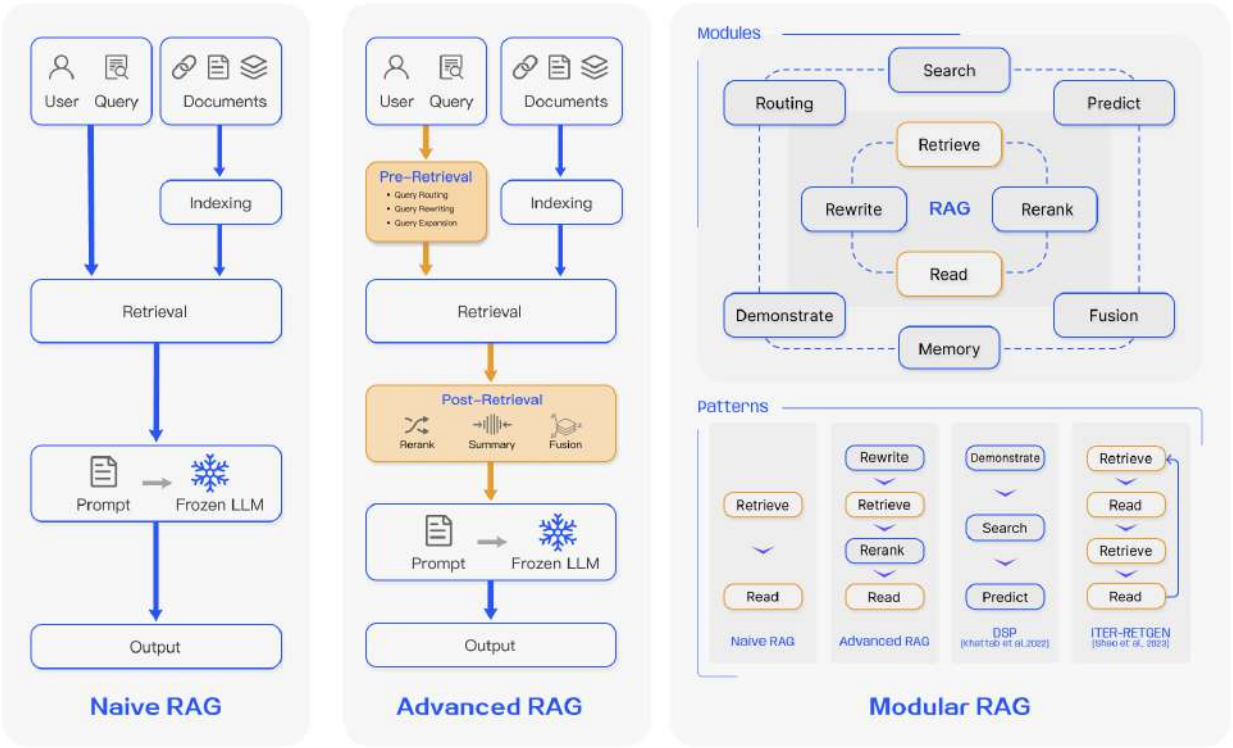


Fig. 3. Comparison between the three paradigms of RAG. (Left) Naive RAG mainly consists of three parts: indexing, retrieval and generation. (Middle) Advanced RAG proposes multiple optimization strategies around pre-retrieval and post-retrieval, with a process similar to the Naive RAG, still following a chain-like structure. (Right) Modular RAG inherits and develops from the previous paradigm, showcasing greater flexibility overall. This is evident in the introduction of multiple specific functional modules and the replacement of existing modules. The overall process is not limited to sequential retrieval and generation; it includes methods such as iterative and adaptive retrieval.

Pre-retrieval process. In this stage, the primary focus is on optimizing the indexing structure and the original query. The goal of optimizing indexing is to enhance the quality of the content being indexed. This involves strategies: enhancing data granularity, optimizing index structures, adding metadata, alignment optimization, and mixed retrieval. While the goal of query optimization is to make the user's original question clearer and more suitable for the retrieval task. Common methods include query rewriting query transformation, query expansion and other techniques [7], [9]–[11].

Post-Retrieval Process. Once relevant context is retrieved, it's crucial to integrate it effectively with the query. The main methods in post-retrieval process include rerank chunks and context compressing. Re-ranking the retrieved information to relocate the most relevant content to the edges of the prompt is a key strategy. This concept has been implemented in frameworks such as LlamaIndex[2], LangChain[3], and HayStack [12]. Feeding all relevant documents directly into LLMs can lead to information overload, diluting the focus on key details with irrelevant content.To mitigate this, post-retrieval efforts concentrate on selecting the essential information, emphasizing critical sections, and shortening the context to be processed.

[2]https://www.llamaindex.ai
[3]https://www.langchain.com/

C. Modular RAG

The modular RAG architecture advances beyond the former two RAG paradigms, offering enhanced adaptability and versatility. It incorporates diverse strategies for improving its components, such as adding a search module for similarity searches and refining the retriever through fine-tuning. Innovations like restructured RAG modules [13] and rearranged RAG pipelines [14] have been introduced to tackle specific challenges. The shift towards a modular RAG approach is becoming prevalent, supporting both sequential processing and integrated end-to-end training across its components. Despite its distinctiveness, Modular RAG builds upon the foundational principles of Advanced and Naive RAG, illustrating a progression and refinement within the RAG family.

1) New Modules: The Modular RAG framework introduces additional specialized components to enhance retrieval and processing capabilities. The Search module adapts to specific scenarios, enabling direct searches across various data sources like search engines, databases, and knowledge graphs, using LLM-generated code and query languages [15]. RAG-Fusion addresses traditional search limitations by employing a multi-query strategy that expands user queries into diverse perspectives, utilizing parallel vector searches and intelligent re-ranking to uncover both explicit and transformative knowledge [16]. The Memory module leverages the LLM's memory to guide retrieval, creating an unbounded memory pool that

aligns the text more closely with data distribution through iterative self-enhancement [17], [18]. Routing in the RAG system navigates through diverse data sources, selecting the optimal pathway for a query, whether it involves summarization, specific database searches, or merging different information streams [19]. The Predict module aims to reduce redundancy and noise by generating context directly through the LLM, ensuring relevance and accuracy [13]. Lastly, the Task Adapter module tailors RAG to various downstream tasks, automating prompt retrieval for zero-shot inputs and creating task-specific retrievers through few-shot query generation [20], [21] .This comprehensive approach not only streamlines the retrieval process but also significantly improves the quality and relevance of the information retrieved, catering to a wide array of tasks and queries with enhanced precision and flexibility.

2) New Patterns: Modular RAG offers remarkable adaptability by allowing module substitution or reconfiguration to address specific challenges. This goes beyond the fixed structures of Naive and Advanced RAG, characterized by a simple "Retrieve" and "Read" mechanism. Moreover, Modular RAG expands this flexibility by integrating new modules or adjusting interaction flow among existing ones, enhancing its applicability across different tasks.

Innovations such as the Rewrite-Retrieve-Read [7]model leverage the LLM's capabilities to refine retrieval queries through a rewriting module and a LM-feedback mechanism to update rewriting model., improving task performance. Similarly, approaches like Generate-Read [13] replace traditional retrieval with LLM-generated content, while Recite-Read [22] emphasizes retrieval from model weights, enhancing the model's ability to handle knowledge-intensive tasks. Hybrid retrieval strategies integrate keyword, semantic, and vector searches to cater to diverse queries. Additionally, employing sub-queries and hypothetical document embeddings (HyDE) [11] seeks to improve retrieval relevance by focusing on embedding similarities between generated answers and real documents.

Adjustments in module arrangement and interaction, such as the Demonstrate-Search-Predict (DSP) [23] framework and the iterative Retrieve-Read-Retrieve-Read flow of ITER-RETGEN [14], showcase the dynamic use of module outputs to bolster another module's functionality, illustrating a sophisticated understanding of enhancing module synergy. The flexible orchestration of Modular RAG Flow showcases the benefits of adaptive retrieval through techniques such as FLARE [24] and Self-RAG [25]. This approach transcends the fixed RAG retrieval process by evaluating the necessity of retrieval based on different scenarios. Another benefit of a flexible architecture is that the RAG system can more easily integrate with other technologies (such as fine-tuning or reinforcement learning) [26]. For example, this can involve fine-tuning the retriever for better retrieval results, fine-tuning the generator for more personalized outputs, or engaging in collaborative fine-tuning [27].

D. RAG vs Fine-tuning

The augmentation of LLMs has attracted considerable attention due to their growing prevalence. Among the optimization methods for LLMs, RAG is often compared with Fine-tuning (FT) and prompt engineering. Each method has distinct characteristics as illustrated in Figure 4. We used a quadrant chart to illustrate the differences among three methods in two dimensions: external knowledge requirements and model adaption requirements. Prompt engineering leverages a model's inherent capabilities with minimum necessity for external knowledge and model adaption. RAG can be likened to providing a model with a tailored textbook for information retrieval, ideal for precise information retrieval tasks. In contrast, FT is comparable to a student internalizing knowledge over time, suitable for scenarios requiring replication of specific structures, styles, or formats.

RAG excels in dynamic environments by offering real-time knowledge updates and effective utilization of external knowledge sources with high interpretability. However, it comes with higher latency and ethical considerations regarding data retrieval. On the other hand, FT is more static, requiring retraining for updates but enabling deep customization of the model's behavior and style. It demands significant computational resources for dataset preparation and training, and while it can reduce hallucinations, it may face challenges with unfamiliar data.

In multiple evaluations of their performance on various knowledge-intensive tasks across different topics, [28] revealed that while unsupervised fine-tuning shows some improvement, RAG consistently outperforms it, for both existing knowledge encountered during training and entirely new knowledge. Additionally, it was found that LLMs struggle to learn new factual information through unsupervised fine-tuning. The choice between RAG and FT depends on the specific needs for data dynamics, customization, and computational capabilities in the application context. RAG and FT are not mutually exclusive and can complement each other, enhancing a model's capabilities at different levels. In some instances, their combined use may lead to optimal performance. The optimization process involving RAG and FT may require multiple iterations to achieve satisfactory results.

III. Retrieval

In the context of RAG, it is crucial to efficiently retrieve relevant documents from the data source. There are several key issues involved, such as the retrieval source, retrieval granularity, pre-processing of the retrieval, and selection of the corresponding embedding model.

A. Retrieval Source

RAG relies on external knowledge to enhance LLMs, while the type of retrieval source and the granularity of retrieval units both affect the final generation results.

1) Data Structure: Initially, text is s the mainstream source of retrieval. Subsequently, the retrieval source expanded to include semi-structured data (PDF) and structured data (Knowledge Graph, KG) for enhancement. In addition to retrieving from original external sources, there is also a growing trend in recent researches towards utilizing content generated by LLMs themselves for retrieval and enhancement purposes.

TABLE I
SUMMARY OF RAG METHODS

Method	Retrieval Source	Retrieval Data Type	Retrieval Granularity	Augmentation Stage	Retrieval process
CoG [29]	Wikipedia	Text	Phrase	Pre-training	Iterative
DenseX [30]	FactoidWiki	Text	Proposition	Inference	Once
EAR [31]	Dataset-base	Text	Sentence	Tuning	Once
UPRISE [20]	Dataset-base	Text	Sentence	Tuning	Once
RAST [32]	Dataset-base	Text	Sentence	Tuning	Once
Self-Mem [17]	Dataset-base	Text	Sentence	Tuning	Iterative
FLARE [24]	Search Engine,Wikipedia	Text	Sentence	Tuning	Adaptive
PGRA [33]	Wikipedia	Text	Sentence	Inference	Once
FILCO [34]	Wikipedia	Text	Sentence	Inference	Once
RADA [35]	Dataset-base	Text	Sentence	Inference	Once
Filter-rerank [36]	Synthesized dataset	Text	Sentence	Inference	Once
R-GQA [37]	Dataset-base	Text	Sentence Pair	Tuning	Once
LLM-R [38]	Dataset-base	Text	Sentence Pair	Inference	Iterative
TIGER [39]	Dataset-base	Text	Item-base	Pre-training	Once
LM-Indexer [40]	Dataset-base	Text	Item-base	Tuning	Once
BEQUE [9]	Dataset-base	Text	Item-base	Tuning	Once
CT-RAG [41]	Synthesized dataset	Text	Item-base	Tuning	Once
Atlas [42]	Wikipedia, Common Crawl	Text	Chunk	Pre-training	Iterative
RAVEN [43]	Wikipedia	Text	Chunk	Pre-training	Once
RETRO++ [44]	Pre-training Corpus	Text	Chunk	Pre-training	Iterative
INSTRUCTRETRO [45]	Pre-training corpus	Text	Chunk	Pre-training	Iterative
RRR [7]	Search Engine	Text	Chunk	Tuning	Once
RA-e2e [46]	Dataset-base	Text	Chunk	Tuning	Once
PROMPTAGATOR [21]	BEIR	Text	Chunk	Tuning	Once
AAR [47]	MSMARCO,Wikipedia	Text	Chunk	Tuning	Once
RA-DIT [27]	Common Crawl,Wikipedia	Text	Chunk	Tuning	Once
RAG-Robust [48]	Wikipedia	Text	Chunk	Tuning	Once
RA-Long-Form [49]	Dataset-base	Text	Chunk	Tuning	Once
CoN [50]	Wikipedia	Text	Chunk	Tuning	Once
Self-RAG [25]	Wikipedia	Text	Chunk	Tuning	Adaptive
BGM [26]	Wikipedia	Text	Chunk	Inference	Once
CoQ [51]	Wikipedia	Text	Chunk	Inference	Iterative
Token-Elimination [52]	Wikipedia	Text	Chunk	Inference	Once
PaperQA [53]	Arxiv,Online Database,PubMed	Text	Chunk	Inference	Iterative
NoiseRAG [54]	FactoidWiki	Text	Chunk	Inference	Once
IAG [55]	Search Engine,Wikipedia	Text	Chunk	Inference	Once
NoMIRACL [56]	Wikipedia	Text	Chunk	Inference	Once
ToC [57]	Search Engine,Wikipedia	Text	Chunk	Inference	Recursive
SKR [58]	Dataset-base,Wikipedia	Text	Chunk	Inference	Adaptive
ITRG [59]	Wikipedia	Text	Chunk	Inference	Iterative
RAG-LongContext [60]	Dataset-base	Text	Chunk	Inference	Once
ITER-RETGEN [14]	Wikipedia	Text	Chunk	Inference	Iterative
IRCoT [61]	Wikipedia	Text	Chunk	Inference	Recursive
LLM-Knowledge-Boundary [62]	Wikipedia	Text	Chunk	Inference	Once
RAPTOR [63]	Dataset-base	Text	Chunk	Inference	Recursive
RECITE [22]	LLMs	Text	Chunk	Inference	Once
ICRALM [64]	Pile,Wikipedia	Text	Chunk	Inference	Iterative
Retrieve-and-Sample [65]	Dataset-base	Text	Doc	Tuning	Once
Zemi [66]	C4	Text	Doc	Tuning	Once
CRAG [67]	Arxiv	Text	Doc	Inference	Once
1-PAGER [68]	Wikipedia	Text	Doc	Inference	Iterative
PRCA [69]	Dataset-base	Text	Doc	Inference	Once
QLM-Doc-ranking [70]	Dataset-base	Text	Doc	Inference	Once
Recomp [71]	Wikipedia	Text	Doc	Inference	Once
DSP [23]	Wikipedia	Text	Doc	Inference	Iterative
RePLUG [72]	Pile	Text	Doc	Inference	Once
ARM-RAG [73]	Dataset-base	Text	Doc	Inference	Iterative
GenRead [13]	LLMs	Text	Doc	Inference	Iterative
UniMS-RAG [74]	Dataset-base	Text	Multi	Tuning	Once
CREA-ICL [19]	Dataset-base	Crosslingual,Text	Sentence	Inference	Once
PKG [75]	LLM	Tabular,Text	Chunk	Inference	Once
SANTA [76]	Dataset-base	Code,Text	Item	Pre-training	Once
SURGE [77]	Freebase	KG	Sub-Graph	Tuning	Once
MK-ToD [78]	Dataset-base	KG	Entity	Tuning	Once
Dual-Feedback-ToD [79]	Dataset-base	KG	Entity Sequence	Tuning	Once
KnowledGPT [15]	Dataset-base	KG	Triplet	Inference	Muti-time
FABULA [80]	Dataset-base,Graph	KG	Entity	Inference	Once
HyKGE [81]	CMeKG	KG	Entity	Inference	Once
KALMV [82]	Wikipedia	KG	Triplet	Inference	Iterative
RoG [83]	Freebase	KG	Triplet	Inference	Iterative
G-Retriever [84]	Dataset-base	TextGraph	Sub-Graph	Inference	Once

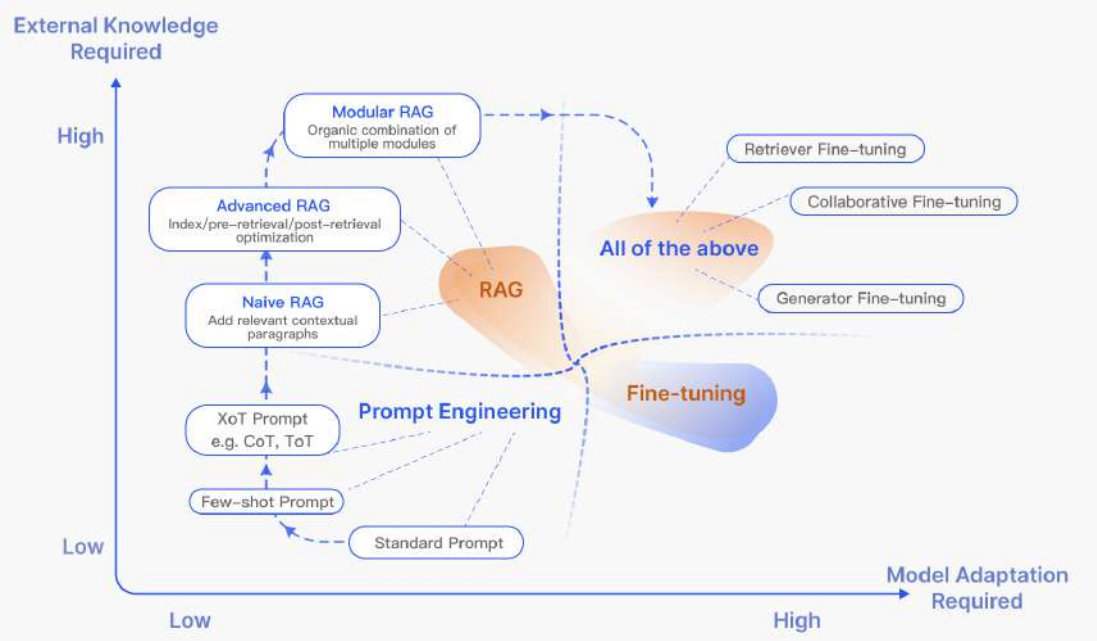


Fig. 4. RAG compared with other model optimization methods in the aspects of "External Knowledge Required" and "Model Adaption Required". Prompt Engineering requires low modifications to the model and external knowledge, focusing on harnessing the capabilities of LLMs themselves. Fine-tuning, on the other hand, involves further training the model. In the early stages of RAG (Naive RAG), there is a low demand for model modifications. As research progresses, Modular RAG has become more integrated with fine-tuning techniques.

Unstructured Data, such as text, is the most widely used retrieval source, which are mainly gathered from corpus. For open-domain question-answering (ODQA) tasks, the primary retrieval sources are Wikipedia Dump with the current major versions including HotpotQA [4] (1st October , 2017), DPR[5] (20 December, 2018). In addition to encyclopedic data, common unstructured data includes cross-lingual text [19] and domain-specific data (such as medical [67]and legal domains [29]).

Semi-structured data. typically refers to data that contains a combination of text and table information, such as PDF. Handling semi-structured data poses challenges for conventional RAG systems due to two main reasons. Firstly, text splitting processes may inadvertently separate tables, leading to data corruption during retrieval. Secondly, incorporating tables into the data can complicate semantic similarity searches. When dealing with semi-structured data, one approach involves leveraging the code capabilities of LLMs to execute Text-2-SQL queries on tables within databases, such as TableGPT [85]. Alternatively, tables can be transformed into text format for further analysis using text-based methods [75]. However, both of these methods are not optimal solutions, indicating substantial research opportunities in this area.

Structured data, such as knowledge graphs (KGs) [86] , which are typically verified and can provide more precise information. KnowledGPT [15] generates KB search queries and stores knowledge in a personalized base, enhancing the RAG model's knowledge richness. In response to the limitations of LLMs in understanding and answering questions about textual graphs, G-Retriever [84] integrates Graph Neural Networks (GNNs), LLMs and RAG, enhancing graph comprehension and question-answering capabilities through soft prompting of the LLM, and employs the Prize-Collecting Steiner Tree (PCST) optimization problem for targeted graph retrieval. On the contrary, it requires additional effort to build, validate, and maintain structured databases. On the contrary, it requires additional effort to build, validate, and maintain structured databases.

LLMs-Generated Content. Addressing the limitations of external auxiliary information in RAG, some research has focused on exploiting LLMs' internal knowledge. SKR [58] classifies questions as known or unknown, applying retrieval enhancement selectively. GenRead [13] replaces the retriever with an LLM generator, finding that LLM-generated contexts often contain more accurate answers due to better alignment with the pre-training objectives of causal language modeling. Selfmem [17] iteratively creates an unbounded memory pool with a retrieval-enhanced generator, using a memory selector to choose outputs that serve as dual problems to the original question, thus self-enhancing the generative model. These methodologies underscore the breadth of innovative data source utilization in RAG, striving to improve model performance and task effectiveness.

2) Retrieval Granularity: Another important factor besides the data format of the retrieval source is the granularity of the retrieved data. Coarse-grained retrieval units theoretically can provide more relevant information for the problem, but they may also contain redundant content, which could distract the retriever and language models in downstream tasks [50], [87]. On the other hand, fine-grained retrieval unit granularity increases the burden of retrieval and does not guarantee semantic integrity and meeting the required knowledge. Choosing

[4]https://hotpotqa.github.io/wiki-readme.html
[5]https://github.com/facebookresearch/DPR

the appropriate retrieval granularity during inference can be a simple and effective strategy to improve the retrieval and downstream task performance of dense retrievers.

In text, retrieval granularity ranges from fine to coarse, including Token, Phrase, Sentence, Proposition, Chunks, Document. Among them, DenseX [30]proposed the concept of using propositions as retrieval units. Propositions are defined as atomic expressions in the text, each encapsulating a unique factual segment and presented in a concise, self-contained natural language format. This approach aims to enhance retrieval precision and relevance. On the Knowledge Graph (KG), retrieval granularity includes Entity, Triplet, and sub-Graph. The granularity of retrieval can also be adapted to downstream tasks, such as retrieving Item IDs [40]in recommendation tasks and Sentence pairs [38]. Detailed information is illustrated in Table II.

B. Indexing Optimization

In the Indexing phase, documents will be processed, segmented, and transformed into Embeddings to be stored in a vector database. The quality of index construction determines whether the correct context can be obtained in the retrieval phase.

1) Chunking Strategy: The most common method is to split the document into chunks on a fixed number of tokens (e.g., 100, 256, 512) [88]. Larger chunks can capture more context, but they also generate more noise, requiring longer processing time and higher costs. While smaller chunks may not fully convey the necessary context, they do have less noise. However, chunks leads to truncation within sentences, prompting the optimization of a recursive splits and sliding window methods, enabling layered retrieval by merging globally related information across multiple retrieval processes [89]. Nevertheless, these approaches still cannot strike a balance between semantic completeness and context length. Therefore, methods like Small2Big have been proposed, where sentences (small) are used as the retrieval unit, and the preceding and following sentences are provided as (big) context to LLMs [90].

2) Metadata Attachments: Chunks can be enriched with metadata information such as page number, file name, author,category timestamp. Subsequently, retrieval can be filtered based on this metadata, limiting the scope of the retrieval. Assigning different weights to document timestamps during retrieval can achieve time-aware RAG, ensuring the freshness of knowledge and avoiding outdated information.

In addition to extracting metadata from the original documents, metadata can also be artificially constructed. For example, adding summaries of paragraph, as well as introducing hypothetical questions. This method is also known as Reverse HyDE. Specifically, using LLM to generate questions that can be answered by the document, then calculating the similarity between the original question and the hypothetical question during retrieval to reduce the semantic gap between the question and the answer.

3) Structural Index: One effective method for enhancing information retrieval is to establish a hierarchical structure for the documents. By constructing In structure, RAG system can expedite the retrieval and processing of pertinent data.

Hierarchical index structure. File are arranged in parent-child relationships, with chunks linked to them. Data summaries are stored at each node, aiding in the swift traversal of data and assisting the RAG system in determining which chunks to extract. This approach can also mitigate the illusion caused by block extraction issues.

Knowledge Graph index. Utilize KG in constructing the hierarchical structure of documents contributes to maintaining consistency. It delineates the connections between different concepts and entities, markedly reducing the potential for illusions. Another advantage is the transformation of the information retrieval process into instructions that LLM can comprehend, thereby enhancing the accuracy of knowledge retrieval and enabling LLM to generate contextually coherent responses, thus improving the overall efficiency of the RAG system. To capture the logical relationship between document content and structure, KGP [91] proposed a method of building an index between multiple documents using KG. This KG consists of nodes (representing paragraphs or structures in the documents, such as pages and tables) and edges (indicating semantic/lexical similarity between paragraphs or relationships within the document structure), effectively addressing knowledge retrieval and reasoning problems in a multi-document environment.

C. Query Optimization

One of the primary challenges with Naive RAG is its direct reliance on the user's original query as the basis for retrieval. Formulating a precise and clear question is difficult, and imprudent queries result in subpar retrieval effectiveness. Sometimes, the question itself is complex, and the language is not well-organized. Another difficulty lies in language complexity ambiguity. Language models often struggle when dealing with specialized vocabulary or ambiguous abbreviations with multiple meanings. For instance, they may not discern whether "LLM" refers to *large language model* or a *Master of Laws* in a legal context.

1) Query Expansion: Expanding a single query into multiple queries enriches the content of the query, providing further context to address any lack of specific nuances, thereby ensuring the optimal relevance of the generated answers.

Multi-Query. By employing prompt engineering to expand queries via LLMs, these queries can then be executed in parallel. The expansion of queries is not random, but rather meticulously designed.

Sub-Query. The process of sub-question planning represents the generation of the necessary sub-questions to contextualize and fully answer the original question when combined. This process of adding relevant context is, in principle, similar to query expansion. Specifically, a complex question can be decomposed into a series of simpler sub-questions using the least-to-most prompting method [92].

Chain-of-Verification(CoVe). The expanded queries undergo validation by LLM to achieve the effect of reducing hallucinations. Validated expanded queries typically exhibit higher reliability [93].

2) Query Transformation: The core concept is to retrieve chunks based on a transformed query instead of the user's original query.

Query Rewrite.The original queries are not always optimal for LLM retrieval, especially in real-world scenarios. Therefore, we can prompt LLM to rewrite the queries. In addition to using LLM for query rewriting, specialized smaller language models, such as RRR (Rewrite-retrieve-read) [7]. The implementation of the query rewrite method in the Taobao, known as BEQUE [9] has notably enhanced recall effectiveness for long-tail queries, resulting in a rise in GMV.

Another query transformation method is to use prompt engineering to let LLM generate a query based on the original query for subsequent retrieval. HyDE [11] construct hypothetical documents (assumed answers to the original query). It focuses on embedding similarity from answer to answer rather than seeking embedding similarity for the problem or query. Using the Step-back Prompting method [10], the original query is abstracted to generate a high-level concept question (step-back question). In the RAG system, both the step-back question and the original query are used for retrieval, and both the results are utilized as the basis for language model answer generation.

3) Query Routing: Based on varying queries, routing to distinct RAG pipeline,which is suitable for a versatile RAG system designed to accommodate diverse scenarios.

Metadata Router/ Filter. The first step involves extracting keywords (entity) from the query, followed by filtering based on the keywords and metadata within the chunks to narrow down the search scope.

Semantic Router is another method of routing involves leveraging the semantic information of the query. Specific apprach see Semantic Router [6]. Certainly, a hybrid routing approach can also be employed, combining both semantic and metadata-based methods for enhanced query routing.

D. Embedding

In RAG, retrieval is achieved by calculating the similarity (e.g. cosine similarity) between the embeddings of the question and document chunks, where the semantic representation capability of embedding models plays a key role. This mainly includes a sparse encoder (BM25) and a dense retriever (BERT architecture Pre-training language models). Recent research has introduced prominent embedding models such as AngIE, Voyage, BGE,etc [94]–[96], which are benefit from multi-task instruct tuning. Hugging Face's MTEB leaderboard [7] evaluates embedding models across 8 tasks, covering 58 datasests. Additionally, C-MTEB focuses on Chinese capability, covering 6 tasks and 35 datasets. There is no one-size-fits-all answer to "which embedding model to use." However, some specific models are better suited for particular use cases.

1) Mix/hybrid Retrieval : Sparse and dense embedding approaches capture different relevance features and can benefit from each other by leveraging complementary relevance information. For instance, sparse retrieval models can be used to provide initial search results for training dense retrieval models. Additionally, pre-training language models (PLMs) can be utilized to learn term weights to enhance sparse retrieval. Specifically, it also demonstrates that sparse retrieval models can enhance the zero-shot retrieval capability of dense retrieval models and assist dense retrievers in handling queries containing rare entities, thereby improving robustness.

2) Fine-tuning Embedding Model: In instances where the context significantly deviates from pre-training corpus, particularly within highly specialized disciplines such as healthcare, legal practice, and other sectors replete with proprietary jargon, fine-tuning the embedding model on your own domain dataset becomes essential to mitigate such discrepancies.

In addition to supplementing domain knowledge, another purpose of fine-tuning is to align the retriever and generator, for example, using the results of LLM as the supervision signal for fine-tuning, known as LSR (LM-supervised Retriever). PROMPTAGATOR [21] utilizes the LLM as a few-shot query generator to create task-specific retrievers, addressing challenges in supervised fine-tuning, particularly in data-scarce domains. Another approach, LLM-Embedder [97], exploits LLMs to generate reward signals across multiple downstream tasks. The retriever is fine-tuned with two types of supervised signals: hard labels for the dataset and soft rewards from the LLMs. This dual-signal approach fosters a more effective fine-tuning process, tailoring the embedding model to diverse downstream applications. REPLUG [72] utilizes a retriever and an LLM to calculate the probability distributions of the retrieved documents and then performs supervised training by computing the KL divergence. This straightforward and effective training method enhances the performance of the retrieval model by using an LM as the supervisory signal, eliminating the need for specific cross-attention mechanisms. Moreover, inspired by RLHF (Reinforcement Learning from Human Feedback), utilizing LM-based feedback to reinforce the retriever through reinforcement learning.

E. Adapter

Fine-tuning models may present challenges, such as integrating functionality through an API or addressing constraints arising from limited local computational resources. Consequently, some approaches opt to incorporate an external adapter to aid in alignment.

To optimize the multi-task capabilities of LLM, UPRISE [20] trained a lightweight prompt retriever that can automatically retrieve prompts from a pre-built prompt pool that are suitable for a given zero-shot task input. AAR (Augmentation-Adapted Retriver) [47] introduces a universal adapter designed to accommodate multiple downstream tasks. While PRCA [69] add a pluggable reward-driven contextual adapter to enhance performance on specific tasks. BGM [26] keeps the retriever and LLM fixed,and trains a bridge Seq2Seq model in between. The bridge model aims to transform the retrieved information into a format that LLMs can work with effectively, allowing it to not only rerank but also dynamically select passages for each query, and potentially employ more advanced strategies like repetition. Furthermore, PKG

[6] https://github.com/aurelio-labs/semantic-router
[7] https://huggingface.co/spaces/mteb/leaderboard

introduces an innovative method for integrating knowledge into white-box models via directive fine-tuning [75]. In this approach, the retriever module is directly substituted to generate relevant documents according to a query. This method assists in addressing the difficulties encountered during the fine-tuning process and enhances model performance.

IV. Generation

After retrieval, it is not a good practice to directly input all the retrieved information to the LLM for answering questions. Following will introduce adjustments from two perspectives: adjusting the retrieved content and adjusting the LLM.

A. Context Curation

Redundant information can interfere with the final generation of LLM, and overly long contexts can also lead LLM to the "Lost in the middle" problem [98]. Like humans, LLM tends to only focus on the beginning and end of long texts, while forgetting the middle portion. Therefore, in the RAG system, we typically need to further process the retrieved content.

1) Reranking: Reranking fundamentally reorders document chunks to highlight the most pertinent results first, effectively reducing the overall document pool, severing a dual purpose in information retrieval, acting as both an enhancer and a filter, delivering refined inputs for more precise language model processing [70]. Reranking can be performed using rule-based methods that depend on predefined metrics like Diversity, Relevance, and MRR, or model-based approaches like Encoder-Decoder models from the BERT series (e.g., SpanBERT), specialized reranking models such as Cohere rerank or bge-raranker-large, and general large language models like GPT [12], [99].

2) Context Selection/Compression: A common misconception in the RAG process is the belief that retrieving as many relevant documents as possible and concatenating them to form a lengthy retrieval prompt is beneficial. However, excessive context can introduce more noise, diminishing the LLM's perception of key information .

(Long) LLMLingua [100], [101] utilize small language models (SLMs) such as GPT-2 Small or LLaMA-7B, to detect and remove unimportant tokens, transforming it into a form that is challenging for humans to comprehend but well understood by LLMs. This approach presents a direct and practical method for prompt compression, eliminating the need for additional training of LLMs while balancing language integrity and compression ratio. PRCA tackled this issue by training an information extractor [69]. Similarly, RECOMP adopts a comparable approach by training an information condenser using contrastive learning [71]. Each training data point consists of one positive sample and five negative samples, and the encoder undergoes training using contrastive loss throughout this process [102] .

In addition to compressing the context, reducing the number of documents aslo helps improve the accuracy of the model's answers. Ma et al. [103] propose the "Filter-Reranker" paradigm, which combines the strengths of LLMs and SLMs. In this paradigm, SLMs serve as filters, while LLMs function as reordering agents. The research shows that instructing LLMs to rearrange challenging samples identified by SLMs leads to significant improvements in various Information Extraction (IE) tasks. Another straightforward and effective approach involves having the LLM evaluate the retrieved content before generating the final answer. This allows the LLM to filter out documents with poor relevance through LLM critique. For instance, in Chatlaw [104], the LLM is prompted to self-suggestion on the referenced legal provisions to assess their relevance.

B. LLM Fine-tuning

Targeted fine-tuning based on the scenario and data characteristics on LLMs can yield better results. This is also one of the greatest advantages of using on-premise LLMs. When LLMs lack data in a specific domain, additional knowledge can be provided to the LLM through fine-tuning. Huggingface's fine-tuning data can also be used as an initial step.

Another benefit of fine-tuning is the ability to adjust the model's input and output. For example, it can enable LLM to adapt to specific data formats and generate responses in a particular style as instructed [37]. For retrieval tasks that engage with structured data, the SANTA framework [76] implements a tripartite training regimen to effectively encapsulate both structural and semantic nuances. The initial phase focuses on the retriever, where contrastive learning is harnessed to refine the query and document embeddings.

Aligning LLM outputs with human or retriever preferences through reinforcement learning is a potential approach. For instance, manually annotating the final generated answers and then providing feedback through reinforcement learning. In addition to aligning with human preferences, it is also possible to align with the preferences of fine-tuned models and retrievers [79]. When circumstances prevent access to powerful proprietary models or larger parameter open-source models, a simple and effective method is to distill the more powerful models(e.g. GPT-4). Fine-tuning of LLM can also be coordinated with fine-tuning of the retriever to align preferences. A typical approach, such as RA-DIT [27], aligns the scoring functions between Retriever and Generator using KL divergence.

V. Augmentation process in RAG

In the domain of RAG, the standard practice often involves a singular (once) retrieval step followed by generation, which can lead to inefficiencies and sometimes is typically insufficient for complex problems demanding multi-step reasoning, as it provides a limited scope of information [105]. Many studies have optimized the retrieval process in response to this issue, and we have summarised them in Figure 5.

A. Iterative Retrieval

Iterative retrieval is a process where the knowledge base is repeatedly searched based on the initial query and the text generated so far, providing a more comprehensive knowledge

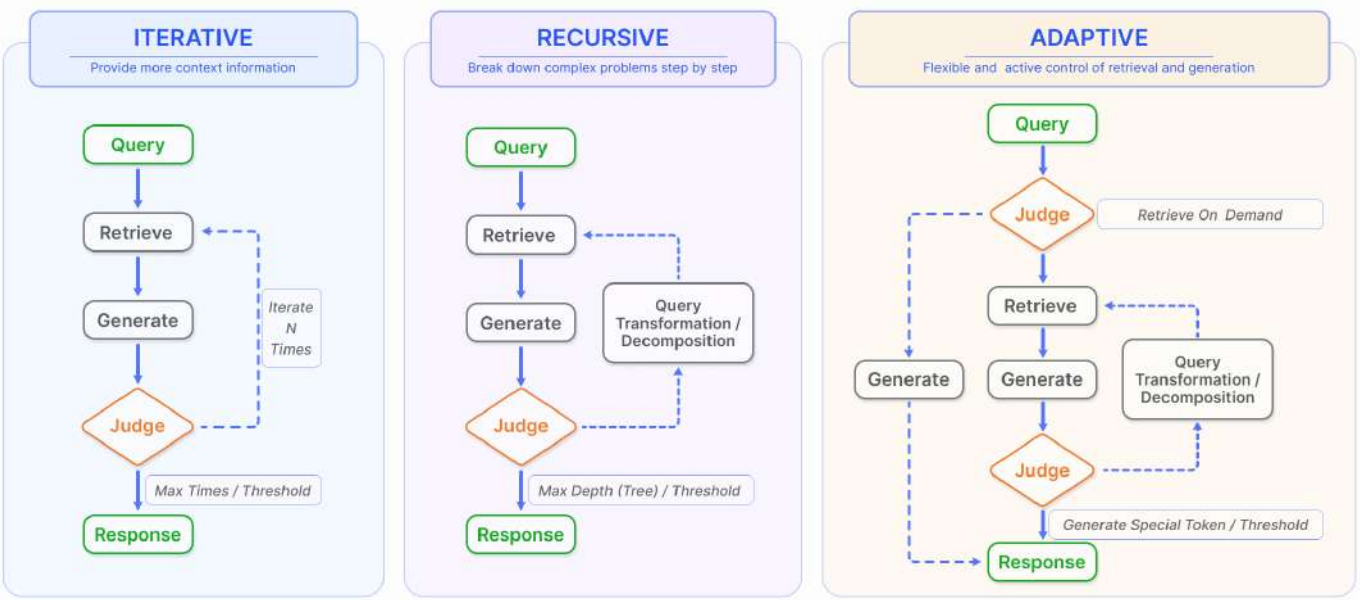


Fig. 5. In addition to the most common once retrieval, RAG also includes three types of retrieval augmentation processes. (left) Iterative retrieval involves alternating between retrieval and generation, allowing for richer and more targeted context from the knowledge base at each step. (Middle) Recursive retrieval involves gradually refining the user query and breaking down the problem into sub-problems, then continuously solving complex problems through retrieval and generation. (Right) Adaptive retrieval focuses on enabling the RAG system to autonomously determine whether external knowledge retrieval is necessary and when to stop retrieval and generation, often utilizing LLM-generated special tokens for control.

base for LLMs. This approach has been shown to enhance the robustness of subsequent answer generation by offering additional contextual references through multiple retrieval iterations. However, it may be affected by semantic discontinuity and the accumulation of irrelevant information. ITER-RETGEN [14] employs a synergistic approach that leverages "retrieval-enhanced generation" alongside "generation-enhanced retrieval" for tasks that necessitate the reproduction of specific information. The model harnesses the content required to address the input task as a contextual basis for retrieving pertinent knowledge, which in turn facilitates the generation of improved responses in subsequent iterations.

B. Recursive Retrieval

Recursive retrieval is often used in information retrieval and NLP to improve the depth and relevance of search results. The process involves iteratively refining search queries based on the results obtained from previous searches. Recursive Retrieval aims to enhance the search experience by gradually converging on the most pertinent information through a feedback loop. IRCoT [61] uses chain-of-thought to guide the retrieval process and refines the CoT with the obtained retrieval results. ToC [57] creates a clarification tree that systematically optimizes the ambiguous parts in the Query. It can be particularly useful in complex search scenarios where the user's needs are not entirely clear from the outset or where the information sought is highly specialized or nuanced. The recursive nature of the process allows for continuous learning and adaptation to the user's requirements, often resulting in improved satisfaction with the search outcomes.

To address specific data scenarios, recursive retrieval and multi-hop retrieval techniques are utilized together. Recursive retrieval involves a structured index to process and retrieve data in a hierarchical manner, which may include summarizing sections of a document or lengthy PDF before performing a retrieval based on this summary. Subsequently, a secondary retrieval within the document refines the search, embodying the recursive nature of the process. In contrast, multi-hop retrieval is designed to delve deeper into graph-structured data sources, extracting interconnected information [106].

C. Adaptive Retrieval

Adaptive retrieval methods, exemplified by Flare [24] and Self-RAG [25], refine the RAG framework by enabling LLMs to actively determine the optimal moments and content for retrieval, thus enhancing the efficiency and relevance of the information sourced.

These methods are part of a broader trend wherein LLMs employ active judgment in their operations, as seen in model agents like AutoGPT, Toolformer, and Graph-Toolformer [107]–[109]. Graph-Toolformer, for instance, divides its retrieval process into distinct steps where LLMs proactively use retrievers, apply Self-Ask techniques, and employ few-shot prompts to initiate search queries. This proactive stance allows LLMs to decide when to search for necessary information, akin to how an agent utilizes tools.

WebGPT [110] integrates a reinforcement learning framework to train the GPT-3 model in autonomously using a search engine during text generation. It navigates this process using special tokens that facilitate actions such as search engine queries, browsing results, and citing references, thereby expanding GPT-3's capabilities through the use of external search engines. Flare automates timing retrieval by monitoring the confidence of the generation process, as indicated by the

probability of generated terms [24]. When the probability falls below a certain threshold would activates the retrieval system to collect relevant information, thus optimizing the retrieval cycle. Self-RAG [25] introduces "reflection tokens" that allow the model to introspect its outputs. These tokens come in two varieties: "retrieve" and "critic". The model autonomously decides when to activate retrieval, or alternatively, a predefined threshold may trigger the process. During retrieval, the generator conducts a fragment-level beam search across multiple paragraphs to derive the most coherent sequence. Critic scores are used to update the subdivision scores, with the flexibility to adjust these weights during inference, tailoring the model's behavior. Self-RAG's design obviates the need for additional classifiers or reliance on Natural Language Inference (NLI) models, thus streamlining the decision-making process for when to engage retrieval mechanisms and improving the model's autonomous judgment capabilities in generating accurate responses.

VI. Task and Evaluation

The rapid advancement and growing adoption of RAG in the field of NLP have propelled the evaluation of RAG models to the forefront of research in the LLMs community. The primary objective of this evaluation is to comprehend and optimize the performance of RAG models across diverse application scenarios.This chapter will mainly introduce the main downstream tasks of RAG, datasets, and how to evaluate RAG systems.

A. Downstream Task

The core task of RAG remains Question Answering (QA), including traditional single-hop/multi-hop QA, multiple-choice, domain-specific QA as well as long-form scenarios suitable for RAG. In addition to QA, RAG is continuously being expanded into multiple downstream tasks, such as Information Extraction (IE), dialogue generation, code search, etc. The main downstream tasks of RAG and their corresponding datasets are summarized in Table III.

B. Evaluation Target

Historically, RAG models assessments have centered on their execution in specific downstream tasks. These evaluations employ established metrics suitable to the tasks at hand. For instance, question answering evaluations might rely on EM and F1 scores [7], [45], [59], [72], whereas fact-checking tasks often hinge on Accuracy as the primary metric [4], [14], [42]. BLEU and ROUGE metrics are also commonly used to evaluate answer quality [26], [32], [52], [78]. Tools like RALLE, designed for the automatic evaluation of RAG applications, similarly base their assessments on these task-specific metrics [160]. Despite this, there is a notable paucity of research dedicated to evaluating the distinct characteristics of RAG models.The main evaluation objectives include:

Retrieval Quality. Evaluating the retrieval quality is crucial for determining the effectiveness of the context sourced by the retriever component. Standard metrics from the domains of search engines, recommendation systems, and information retrieval systems are employed to measure the performance of the RAG retrieval module. Metrics such as Hit Rate, MRR, and NDCG are commonly utilized for this purpose [161], [162].

Generation Quality. The assessment of generation quality centers on the generator's capacity to synthesize coherent and relevant answers from the retrieved context. This evaluation can be categorized based on the content's objectives: unlabeled and labeled content. For unlabeled content, the evaluation encompasses the faithfulness, relevance, and non-harmfulness of the generated answers. In contrast, for labeled content, the focus is on the accuracy of the information produced by the model [161]. Additionally, both retrieval and generation quality assessments can be conducted through manual or automatic evaluation methods [29], [161], [163].

C. Evaluation Aspects

Contemporary evaluation practices of RAG models emphasize three primary quality scores and four essential abilities, which collectively inform the evaluation of the two principal targets of the RAG model: retrieval and generation.

1) Quality Scores: Quality scores include context relevance, answer faithfulness, and answer relevance. These quality scores evaluate the efficiency of the RAG model from different perspectives in the process of information retrieval and generation [164]–[166].

Context Relevance evaluates the precision and specificity of the retrieved context, ensuring relevance and minimizing processing costs associated with extraneous content.

Answer Faithfulness ensures that the generated answers remain true to the retrieved context, maintaining consistency and avoiding contradictions.

Answer Relevance requires that the generated answers are directly pertinent to the posed questions, effectively addressing the core inquiry.

2) Required Abilities: RAG evaluation also encompasses four abilities indicative of its adaptability and efficiency: noise robustness, negative rejection, information integration, and counterfactual robustness [167], [168]. These abilities are critical for the model's performance under various challenges and complex scenarios, impacting the quality scores.

Noise Robustness appraises the model's capability to manage noise documents that are question-related but lack substantive information.

Negative Rejection assesses the model's discernment in refraining from responding when the retrieved documents do not contain the necessary knowledge to answer a question.

Information Integration evaluates the model's proficiency in synthesizing information from multiple documents to address complex questions.

Counterfactual Robustness tests the model's ability to recognize and disregard known inaccuracies within documents, even when instructed about potential misinformation.

Context relevance and noise robustness are important for evaluating the quality of retrieval, while answer faithfulness, answer relevance, negative rejection, information integration, and counterfactual robustness are important for evaluating the quality of generation.

TABLE II
DOWNSTREAM TASKS AND DATASETS OF RAG

Task	Sub Task	Dataset	Method
QA	Single-hop	Natural Qustion(NQ) [111]	[26], [30], [34], [42], [45], [50], [52], [59], [64], [82] [3], [4], [22], [27], [40], [43], [54], [62], [71], [112] [20], [44], [72]
		TriviaQA(TQA) [113]	[13], [30], [34], [45], [50], [64] [4], [27], [59], [62], [112] [22], [25], [43], [44], [71], [72]
		SQuAD [114]	[20], [23], [30], [32], [45], [69], [112]
		Web Questions(WebQ) [115]	[3], [4], [13], [30], [50], [68]
		PopQA [116]	[7], [25], [67]
		MS MARCO [117]	[4], [40], [52]
	Multi-hop	HotpotQA [118]	[23], [26], [31], [34], [47], [51], [61], [82] [7], [14], [22], [27], [59], [62], [69], [71], [91]
		2WikiMultiHopQA [119]	[14], [24], [48], [59], [61], [91]
		MuSiQue [120]	[14], [51], [61], [91]
	Long-form QA	ELI5 [121]	[27], [34], [43], [49], [51]
		NarrativeQA(NQA) [122]	[45], [60], [63], [123]
		ASQA [124]	[24], [57]
		QMSum(QM) [125]	[60], [123]
	Domain QA	Qasper [126]	[60], [63]
		COVID-QA [127]	[35], [46]
		CMB [128],MMCU_Medical [129]	[81]
	Multi-Choice QA	QuALITY [130]	[60], [63]
		ARC [131]	[25], [67]
		CommonsenseQA [132]	[58], [66]
	Graph QA	GraphQA [84]	[84]
Dialog	Dialog Generation	Wizard of Wikipedia (WoW) [133]	[13], [27], [34], [42]
	Personal Dialog	KBP [134]	[74], [135]
		DuleMon [136]	[74]
	Task-oriented Dialog	CamRest [137]	[78], [79]
	Recommendation	Amazon(Toys,Sport,Beauty) [138]	[39], [40]
IE	Event Argument Extraction	WikiEvent [139]	[13], [27], [37], [42]
		RAMS [140]	[36], [37]
	Relation Extraction	T-REx [141],ZsRE [142]	[27], [51]
Reasoning	Commonsense Reasoning	HellaSwag [143]	[20], [66]
	CoT Reasoning	CoT Reasoning [144]	[27]
	Complex Reasoning	CSQA [145]	[55]
Others	Language Understanding	MMLU [146]	[7], [27], [28], [42], [43], [47], [72]
	Language Modeling	WikiText-103 [147]	[5], [29], [64], [71]
		StrategyQA [148]	[14], [24], [48], [51], [55], [58]
	Fact Checking/Verification	FEVER [149]	[4], [13], [27], [34], [42], [50]
		PubHealth [150]	[25], [67]
	Text Generation	Biography [151]	[67]
	Text Summarization	WikiASP [152]	[24]
		XSum [153]	[17]
	Text Classification	VioLens [154]	[19]
		TREC [155]	[33]
	Sentiment	SST-2 [156]	[20], [33], [38]
	Code Search	CodeSearchNet [157]	[76]
	Robustness Evaluation	NoMIRACL [56]	[56]
	Math	GSM8K [158]	[73]
	Machine Translation	JRC-Acquis [159]	[17]

TABLE III
SUMMARY OF METRICS APPLICABLE FOR EVALUATION ASPECTS OF RAG

	Context Relevance	Faithfulness	Answer Relevance	Noise Robustness	Negative Rejection	Information Integration	Counterfactual Robustness
Accuracy	✓	✓	✓	✓	✓	✓	✓
EM					✓		
Recall	✓						
Precision	✓			✓			
R-Rate							✓
Cosine Similarity			✓				
Hit Rate	✓						
MRR	✓						
NDCG	✓						
BLEU	✓	✓	✓				
ROUGE/ROUGE-L	✓	✓	✓				

The specific metrics for each evaluation aspect are summarized in Table III. It is essential to recognize that these metrics, derived from related work, are traditional measures and do not yet represent a mature or standardized approach for quantifying RAG evaluation aspects. Custom metrics tailored to the nuances of RAG models, though not included here, have also been developed in some evaluation studies.

D. Evaluation Benchmarks and Tools

A series of benchmark tests and tools have been proposed to facilitate the evaluation of RAG.These instruments furnish quantitative metrics that not only gauge RAG model performance but also enhance comprehension of the model's capabilities across various evaluation aspects. Prominent benchmarks such as RGB, RECALL and CRUD [167]–[169] focus on appraising the essential abilities of RAG models. Concurrently, state-of-the-art automated tools like RAGAS [164], ARES [165], and TruLens[8] employ LLMs to adjudicate the quality scores. These tools and benchmarks collectively form a robust framework for the systematic evaluation of RAG models, as summarized in Table IV.

VII. DISCUSSION AND FUTURE PROSPECTS

Despite the considerable progress in RAG technology, several challenges persist that warrant in-depth research.This chapter will mainly introduce the current challenges and future research directions faced by RAG.

A. RAG vs Long Context

With the deepening of related research, the context of LLMs is continuously expanding [170]–[172]. Presently, LLMs can effortlessly manage contexts exceeding 200,000 tokens [9]. This capability signifies that long-document question answering, previously reliant on RAG, can now incorporate the entire document directly into the prompt. This has also sparked discussions on whether RAG is still necessary when LLMs are not constrained by context. In fact, RAG still plays an irreplaceable role. On one hand, providing LLMs with a large amount of context at once will significantly impact its inference speed, while chunked retrieval and on-demand input can significantly improve operational efficiency. On the other hand, RAG-based generation can quickly locate the original references for LLMs to help users verify the generated answers. The entire retrieval and reasoning process is observable, while generation solely relying on long context remains a black box. Conversely, the expansion of context provides new opportunities for the development of RAG, enabling it to address more complex problems and integrative or summary questions that require reading a large amount of material to answer [49]. Developing new RAG methods in the context of super-long contexts is one of the future research trends.

B. RAG Robustness

The presence of noise or contradictory information during retrieval can detrimentally affect RAG's output quality. This situation is figuratively referred to as "Misinformation can be worse than no information at all". Improving RAG's resistance to such adversarial or counterfactual inputs is gaining research momentum and has become a key performance metric [48], [50], [82]. Cuconasu et al. [54] analyze which type of documents should be retrieved, evaluate the relevance of the documents to the prompt, their position, and the number included in the context. The research findings reveal that including irrelevant documents can unexpectedly increase accuracy by over 30%, contradicting the initial assumption of reduced quality. These results underscore the importance of developing specialized strategies to integrate retrieval with language generation models, highlighting the need for further research and exploration into the robustness of RAG.

C. Hybrid Approaches

Combining RAG with fine-tuning is emerging as a leading strategy. Determining the optimal integration of RAG and fine-tuning whether sequential, alternating, or through end-to-end joint training—and how to harness both parameterized

[8] https://www.trulens.org/trulens_eval/core_concepts_rag_triad/
[9] https://kimi.moonshot.cn

TABLE IV
SUMMARY OF EVALUATION FRAMEWORKS

Evaluation Framework	Evaluation Targets	Evaluation Aspects	Quantitative Metrics
RGB†	Retrieval Quality	Noise Robustness	Accuracy
		Negative Rejection	EM
	Generation Quality	Information Integration	Accuracy
		Counterfactual Robustness	Accuracy
RECALL†	Generation Quality	Counterfactual Robustness	R-Rate (Reappearance Rate)
RAGAS‡	Retrieval Quality	Context Relevance	*
	Generation Quality	Faithfulness	*
		Answer Relevance	Cosine Similarity
ARES‡	Retrieval Quality	Context Relevance	Accuracy
	Generation Quality	Faithfulness	Accuracy
		Answer Relevance	Accuracy
TruLens‡	Retrieval Quality	Context Relevance	*
	Generation Quality	Faithfulness	*
		Answer Relevance	*
CRUD†	Retrieval Quality	Creative Generation	BLEU
		Knowledge-intensive QA	ROUGE-L
	Generation Quality	Error Correction	BertScore
		Summarization	RAGQuestEval

*† represents a benchmark, and ‡ represents a tool. * denotes customized quantitative metrics, which deviate from traditional metrics. Readers are encouraged to consult pertinent literature for the specific quantification formulas associated with these metrics, as required.*

and non-parameterized advantages are areas ripe for exploration [27]. Another trend is to introduce SLMs with specific functionalities into RAG and fine-tuned by the results of RAG system. For example, CRAG [67] trains a lightweight retrieval evaluator to assess the overall quality of the retrieved documents for a query and triggers different knowledge retrieval actions based on confidence levels.

D. Scaling laws of RAG

End-to-end RAG models and pre-trained models based on RAG are still one of the focuses of current researchers [173].The parameters of these models are one of the key factors.While scaling laws [174] are established for LLMs, their applicability to RAG remains uncertain. Initial studies like RETRO++ [44] have begun to address this, yet the parameter count in RAG models still lags behind that of LLMs. The possibility of an Inverse Scaling Law [10], where smaller models outperform larger ones, is particularly intriguing and merits further investigation.

E. Production-Ready RAG

RAG's practicality and alignment with engineering requirements have facilitated its adoption. However, enhancing retrieval efficiency, improving document recall in large knowledge bases, and ensuring data security—such as preventing inadvertent disclosure of document sources or metadata by LLMs—are critical engineering challenges that remain to be addressed [175].

The development of the RAG ecosystem is greatly impacted by the progression of its technical stack. Key tools like LangChain and LLamaIndex have quickly gained popularity with the emergence of ChatGPT, providing extensive RAG-related APIs and becoming essential in the realm of LLMs.The emerging technology stack, while not as rich in features as LangChain and LLamaIndex, stands out through its specialized products. For example, Flowise AI prioritizes a low-code approach, allowing users to deploy AI applications, including RAG, through a user-friendly drag-and-drop interface. Other technologies like HayStack, Meltano, and Cohere Coral are also gaining attention for their unique contributions to the field.

In addition to AI-focused vendors, traditional software and cloud service providers are expanding their offerings to include RAG-centric services. Weaviate's Verba [11] is designed for personal assistant applications, while Amazon's Kendra [12] offers intelligent enterprise search services, enabling users to browse various content repositories using built-in connectors. In the development of RAG technology, there is a clear trend towards different specialization directions, such as: 1) Customization - tailoring RAG to meet specific requirements. 2) Simplification - making RAG easier to use to reduce the

[10] https://github.com/inverse-scaling/prize

[11] https://github.com/weaviate/Verba

[12] https://aws.amazon.com/cn/kendra/

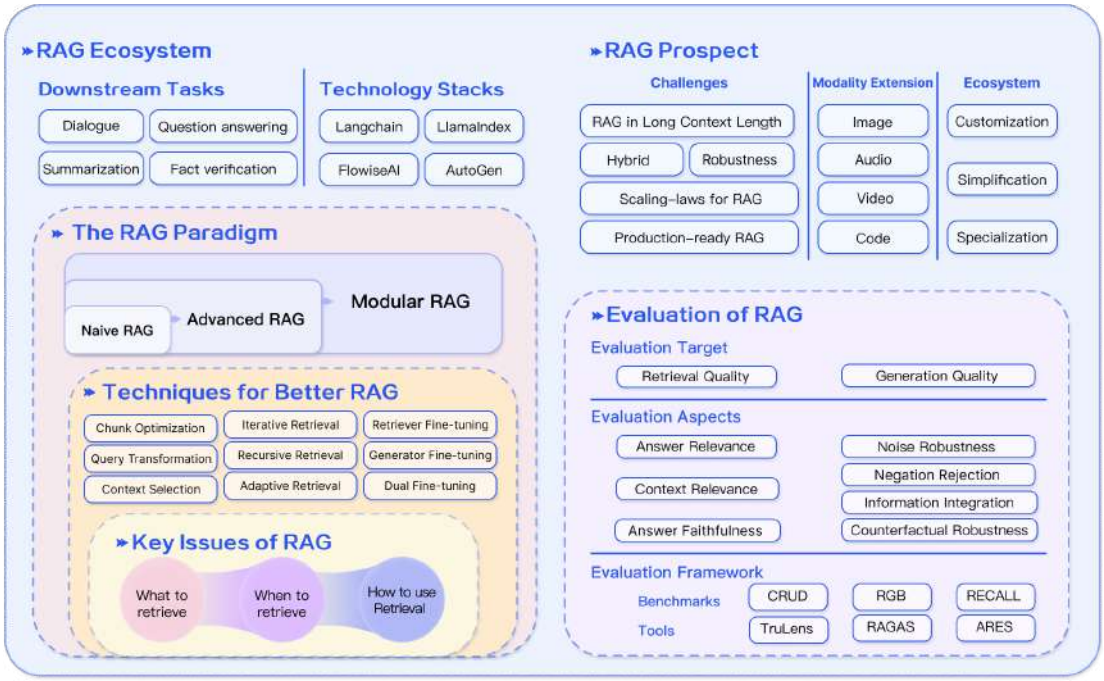


Fig. 6. Summary of RAG ecosystem

initial learning curve. 3) Specialization - optimizing RAG to better serve production environments.

The mutual growth of RAG models and their technology stacks is evident; technological advancements continuously establish new standards for existing infrastructure. In turn, enhancements to the technology stack drive the development of RAG capabilities. RAG toolkits are converging into a foundational technology stack, laying the groundwork for advanced enterprise applications. However, a fully integrated, comprehensive platform concept is still in the future, requiring further innovation and development.

F. Multi-modal RAG

RAG has transcended its initial text-based question-answering confines, embracing a diverse array of modal data. This expansion has spawned innovative multimodal models that integrate RAG concepts across various domains:

Image. RA-CM3 [176] stands as a pioneering multimodal model of both retrieving and generating text and images. BLIP-2 [177] leverages frozen image encoders alongside LLMs for efficient visual language pre-training, enabling zero-shot image-to-text conversions. The "Visualize Before You Write" method [178] employs image generation to steer the LM's text generation, showing promise in open-ended text generation tasks.

Audio and Video. The GSS method retrieves and stitches together audio clips to convert machine-translated data into speech-translated data [179]. UEOP marks a significant advancement in end-to-end automatic speech recognition by incorporating external, offline strategies for voice-to-text conversion [180]. Additionally, KNN-based attention fusion leverages audio embeddings and semantically related text embeddings to refine ASR, thereby accelerating domain adaptation. Vid2Seq augments language models with specialized temporal markers, facilitating the prediction of event boundaries and textual descriptions within a unified output sequence [181].

Code. RBPS [182] excels in small-scale learning tasks by retrieving code examples that align with developers' objectives through encoding and frequency analysis. This approach has demonstrated efficacy in tasks such as test assertion generation and program repair. For structured knowledge, the CoK method [106] first extracts facts pertinent to the input query from a knowledge graph, then integrates these facts as hints within the input, enhancing performance in knowledge graph question-answering tasks.

VIII. Conclusion

The summary of this paper, as depicted in Figure 6, emphasizes RAG's significant advancement in enhancing the capabilities of LLMs by integrating parameterized knowledge from language models with extensive non-parameterized data from external knowledge bases. The survey showcases the evolution of RAG technologies and their application on many different tasks. The analysis outlines three developmental paradigms within the RAG framework: Naive, Advanced, and Modular RAG, each representing a progressive enhancement over its predecessors. RAG's technical integration with other AI methodologies, such as fine-tuning and reinforcement learning, has further expanded its capabilities. Despite the progress in RAG technology, there are research opportunities to improve its robustness and its ability to handle extended contexts. RAG's application scope is expanding into multimodal domains, adapting its principles to interpret and process diverse data forms like images, videos, and code. This expansion highlights RAG's significant practical implications for AI deployment, attracting interest from academic and industrial sectors.

The growing ecosystem of RAG is evidenced by the rise in RAG-centric AI applications and the continuous development of supportive tools. As RAG's application landscape broadens, there is a need to refine evaluation methodologies to keep pace with its evolution. Ensuring accurate and representative performance assessments is crucial for fully capturing RAG's contributions to the AI research and development community.

REFERENCES

[1] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 696–15 707.

[2] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen *et al.*, "Siren's song in the ai ocean: A survey on hallucination in large language models," *arXiv preprint arXiv:2309.01219*, 2023.

[3] D. Arora, A. Kini, S. R. Chowdhury, N. Natarajan, G. Sinha, and A. Sharma, "Gar-meets-rag paradigm for zero-shot information retrieval," *arXiv preprint arXiv:2310.20158*, 2023.

[4] 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.

[5] 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 *International conference on machine learning*. PMLR, 2022, pp. 2206–2240.

[6] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[7] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, "Query rewriting for retrieval-augmented large language models," *arXiv preprint arXiv:2305.14283*, 2023.

[8] I. ILIN, "Advanced rag techniques: an illustrated overview," https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6, 2023.

[9] W. Peng, G. Li, Y. Jiang, Z. Wang, D. Ou, X. Zeng, E. Chen *et al.*, "Large language model based long-tail query rewriting in taobao search," *arXiv preprint arXiv:2311.03758*, 2023.

[10] H. S. Zheng, S. Mishra, X. Chen, H.-T. Cheng, E. H. Chi, Q. V. Le, and D. Zhou, "Take a step back: Evoking reasoning via abstraction in large language models," *arXiv preprint arXiv:2310.06117*, 2023.

[11] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," *arXiv preprint arXiv:2212.10496*, 2022.

[12] V. Blagojevi, "Enhancing rag pipelines in haystack: Introducing diversityranker and lostinthemiddleranker," https://towardsdatascience.com/enhancing-rag-pipelines-in-haystack-45f14e2bc9f5, 2023.

[13] W. Yu, D. Iter, S. Wang, Y. Xu, M. Ju, S. Sanyal, C. Zhu, M. Zeng, and M. Jiang, "Generate rather than retrieve: Large language models are strong context generators," *arXiv preprint arXiv:2209.10063*, 2022.

[14] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," *arXiv preprint arXiv:2305.15294*, 2023.

[15] X. Wang, Q. Yang, Y. Qiu, J. Liang, Q. He, Z. Gu, Y. Xiao, and W. Wang, "Knowledgpt: Enhancing large language models with retrieval and storage access on knowledge bases," *arXiv preprint arXiv:2308.11761*, 2023.

[16] A. H. Raudaschl, "Forget rag, the future is rag-fusion," https://towardsdatascience.com/forget-rag-the-future-is-rag-fusion-1147298d8ad1, 2023.

[17] X. Cheng, D. Luo, X. Chen, L. Liu, D. Zhao, and R. Yan, "Lift yourself up: Retrieval-augmented text generation with self memory," *arXiv preprint arXiv:2305.02437*, 2023.

[18] S. Wang, Y. Xu, Y. Fang, Y. Liu, S. Sun, R. Xu, C. Zhu, and M. Zeng, "Training data is more valuable than you think: A simple and effective method by retrieving from training data," *arXiv preprint arXiv:2203.08773*, 2022.

[19] X. Li, E. Nie, and S. Liang, "From classification to generation: Insights into crosslingual retrieval augmented icl," *arXiv preprint arXiv:2311.06595*, 2023.

[20] D. Cheng, S. Huang, J. Bi, Y. Zhan, J. Liu, Y. Wang, H. Sun, F. Wei, D. Deng, and Q. Zhang, "Uprise: Universal prompt retrieval for improving zero-shot evaluation," *arXiv preprint arXiv:2303.08518*, 2023.

[21] Z. Dai, V. Y. Zhao, J. Ma, Y. Luan, J. Ni, J. Lu, A. Bakalov, K. Guu, K. B. Hall, and M.-W. Chang, "Promptagator: Few-shot dense retrieval from 8 examples," *arXiv preprint arXiv:2209.11755*, 2022.

[22] Z. Sun, X. Wang, Y. Tay, Y. Yang, and D. Zhou, "Recitation-augmented language models," *arXiv preprint arXiv:2210.01296*, 2022.

[23] 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 arXiv:2212.14024*, 2022.

[24] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," *arXiv preprint arXiv:2305.06983*, 2023.

[25] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-rag: Learning to retrieve, generate, and critique through self-reflection," *arXiv preprint arXiv:2310.11511*, 2023.

[26] Z. Ke, W. Kong, C. Li, M. Zhang, Q. Mei, and M. Bendersky, "Bridging the preference gap between retrievers and llms," *arXiv preprint arXiv:2401.06954*, 2024.

[27] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, R. James, P. Rodriguez, J. Kahn, G. Szilvasy, M. Lewis *et al.*, "Ra-dit: Retrieval-augmented dual instruction tuning," *arXiv preprint arXiv:2310.01352*, 2023.

[28] O. Ovadia, M. Brief, M. Mishaeli, and O. Elisha, "Fine-tuning or retrieval? comparing knowledge injection in llms," *arXiv preprint arXiv:2312.05934*, 2023.

[29] T. Lan, D. Cai, Y. Wang, H. Huang, and X.-L. Mao, "Copy is all you need," in *The Eleventh International Conference on Learning Representations*, 2022.

[30] T. Chen, H. Wang, S. Chen, W. Yu, K. Ma, X. Zhao, D. Yu, and H. Zhang, "Dense x retrieval: What retrieval granularity should we use?" *arXiv preprint arXiv:2312.06648*, 2023.

[31] F. Luo and M. Surdeanu, "Divide & conquer for entailment-aware multi-hop evidence retrieval," *arXiv preprint arXiv:2311.02616*, 2023.

[32] Q. Gou, Z. Xia, B. Yu, H. Yu, F. Huang, Y. Li, and N. Cam-Tu, "Diversify question generation with retrieval-augmented style transfer," *arXiv preprint arXiv:2310.14503*, 2023.

[33] Z. Guo, S. Cheng, Y. Wang, P. Li, and Y. Liu, "Prompt-guided retrieval augmentation for non-knowledge-intensive tasks," *arXiv preprint arXiv:2305.17653*, 2023.

[34] Z. Wang, J. Araki, Z. Jiang, M. R. Parvez, and G. Neubig, "Learning to filter context for retrieval-augmented generation," *arXiv preprint arXiv:2311.08377*, 2023.

[35] M. Seo, J. Baek, J. Thorne, and S. J. Hwang, "Retrieval-augmented data augmentation for low-resource domain tasks," *arXiv preprint arXiv:2402.13482*, 2024.

[36] Y. Ma, Y. Cao, Y. Hong, and A. Sun, "Large language model is not a good few-shot information extractor, but a good reranker for hard samples!" *arXiv preprint arXiv:2303.08559*, 2023.

[37] X. Du and H. Ji, "Retrieval-augmented generative question answering for event argument extraction," *arXiv preprint arXiv:2211.07067*, 2022.

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

[39] S. Rajput, N. Mehta, A. Singh, R. H. Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Q. Tran, J. Samost *et al.*, "Recommender systems with generative retrieval," *arXiv preprint arXiv:2305.05065*, 2023.

[40] B. Jin, H. Zeng, G. Wang, X. Chen, T. Wei, R. Li, Z. Wang, Z. Li, Y. Li, H. Lu *et al.*, "Language models as semantic indexers," *arXiv preprint arXiv:2310.07815*, 2023.

[41] R. Anantha, T. Bethi, D. Vodianik, and S. Chappidi, "Context tuning for retrieval augmented generation," *arXiv preprint arXiv:2312.05708*, 2023.

[42] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Few-shot learning with retrieval augmented language models," *arXiv preprint arXiv:2208.03299*, 2022.

[43] J. Huang, W. Ping, P. Xu, M. Shoeybi, K. C.-C. Chang, and B. Catanzaro, "Raven: In-context learning with retrieval augmented encoder-decoder language models," *arXiv preprint arXiv:2308.07922*, 2023.

[44] B. Wang, W. Ping, P. Xu, L. McAfee, Z. Liu, M. Shoeybi, Y. Dong, O. Kuchaiev, B. Li, C. Xiao *et al.*, "Shall we pretrain autoregressive language models with retrieval? a comprehensive study," *arXiv preprint arXiv:2304.06762*, 2023.

[45] B. Wang, W. Ping, L. McAfee, P. Xu, B. Li, M. Shoeybi, and B. Catanzaro, "Instructretro: Instruction tuning post retrieval-augmented pretraining," *arXiv preprint arXiv:2310.07713*, 2023.

[46] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1–17, 2023.

[47] Z. Yu, C. Xiong, S. Yu, and Z. Liu, "Augmentation-adapted retriever improves generalization of language models as generic plug-in," *arXiv preprint arXiv:2305.17331*, 2023.

[48] O. Yoran, T. Wolfson, O. Ram, and J. Berant, "Making retrieval-augmented language models robust to irrelevant context," *arXiv preprint arXiv:2310.01558*, 2023.

[49] H.-T. Chen, F. Xu, S. A. Arora, and E. Choi, "Understanding retrieval augmentation for long-form question answering," *arXiv preprint arXiv:2310.12150*, 2023.

[50] W. Yu, H. Zhang, X. Pan, K. Ma, H. Wang, and D. Yu, "Chain-of-note: Enhancing robustness in retrieval-augmented language models," *arXiv preprint arXiv:2311.09210*, 2023.

[51] S. Xu, L. Pang, H. Shen, X. Cheng, and T.-S. Chua, "Search-in-the-chain: Towards accurate, credible and traceable large language models for knowledgeintensive tasks," *CoRR, vol. abs/2304.14732*, 2023.

[52] M. Berchansky, P. Izsak, A. Caciularu, I. Dagan, and M. Wasserblat, "Optimizing retrieval-augmented reader models via token elimination," *arXiv preprint arXiv:2310.13682*, 2023.

[53] 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 arXiv:2312.07559*, 2023.

[54] F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri, "The power of noise: Redefining retrieval for rag systems," *arXiv preprint arXiv:2401.14887*, 2024.

[55] Z. Zhang, X. Zhang, Y. Ren, S. Shi, M. Han, Y. Wu, R. Lai, and Z. Cao, "Iag: Induction-augmented generation framework for answering reasoning questions," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 1–14.

[56] N. Thakur, L. Bonifacio, X. Zhang, O. Ogundepo, E. Kamalloo, D. Alfonso-Hermelo, X. Li, Q. Liu, B. Chen, M. Rezagholizadeh *et al.*, "Nomiracl: Knowing when you don't know for robust multilingual retrieval-augmented generation," *arXiv preprint arXiv:2312.11361*, 2023.

[57] G. Kim, S. Kim, B. Jeon, J. Park, and J. Kang, "Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models," *arXiv preprint arXiv:2310.14696*, 2023.

[58] Y. Wang, P. Li, M. Sun, and Y. Liu, "Self-knowledge guided retrieval augmentation for large language models," *arXiv preprint arXiv:2310.05002*, 2023.

[59] Z. Feng, X. Feng, D. Zhao, M. Yang, and B. Qin, "Retrieval-generation synergy augmented large language models," *arXiv preprint arXiv:2310.05149*, 2023.

[60] P. Xu, W. Ping, X. Wu, L. McAfee, C. Zhu, Z. Liu, S. Subramanian, E. Bakhturina, M. Shoeybi, and B. Catanzaro, "Retrieval meets long context large language models," *arXiv preprint arXiv:2310.03025*, 2023.

[61] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," *arXiv preprint arXiv:2212.10509*, 2022.

[62] R. Ren, Y. Wang, Y. Qu, W. X. Zhao, J. Liu, H. Tian, H. Wu, J.-R. Wen, and H. Wang, "Investigating the factual knowledge boundary of large language models with retrieval augmentation," *arXiv preprint arXiv:2307.11019*, 2023.

[63] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," *arXiv preprint arXiv:2401.18059*, 2024.

[64] O. Ram, Y. Levine, I. Dalmedigos, D. Muhlgay, A. Shashua, K. Leyton-Brown, and Y. Shoham, "In-context retrieval-augmented language models," *arXiv preprint arXiv:2302.00083*, 2023.

[65] Y. Ren, Y. Cao, P. Guo, F. Fang, W. Ma, and Z. Lin, "Retrieve-and-sample: Document-level event argument extraction via hybrid retrieval augmentation," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 293–306.

[66] Z. Wang, X. Pan, D. Yu, D. Yu, J. Chen, and H. Ji, "Zemi: Learning zero-shot semi-parametric language models from multiple tasks," *arXiv preprint arXiv:2210.00185*, 2022.

[67] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, "Corrective retrieval augmented generation," *arXiv preprint arXiv:2401.15884*, 2024.

[68] P. Jain, L. B. Soares, and T. Kwiatkowski, "1-pager: One pass answer generation and evidence retrieval," *arXiv preprint arXiv:2310.16568*, 2023.

[69] 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," *arXiv preprint arXiv:2310.18347*, 2023.

[70] S. Zhuang, B. Liu, B. Koopman, and G. Zuccon, "Open-source large language models are strong zero-shot query likelihood models for document ranking," *arXiv preprint arXiv:2310.13243*, 2023.

[71] F. Xu, W. Shi, and E. Choi, "Recomp: Improving retrieval-augmented lms with compression and selective augmentation," *arXiv preprint arXiv:2310.04408*, 2023.

[72] W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W.-t. Yih, "Replug: Retrieval-augmented black-box language models," *arXiv preprint arXiv:2301.12652*, 2023.

[73] E. Melz, "Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation," *arXiv preprint arXiv:2311.04177*, 2023.

[74] H. Wang, W. Huang, Y. Deng, R. Wang, Z. Wang, Y. Wang, F. Mi, J. Z. Pan, and K.-F. Wong, "Unims-rag: A unified multi-source retrieval-augmented generation for personalized dialogue systems," *arXiv preprint arXiv:2401.13256*, 2024.

[75] Z. Luo, C. Xu, P. Zhao, X. Geng, C. Tao, J. Ma, Q. Lin, and D. Jiang, "Augmented large language models with parametric knowledge guiding," *arXiv preprint arXiv:2305.04757*, 2023.

[76] X. Li, Z. Liu, C. Xiong, S. Yu, Y. Gu, Z. Liu, and G. Yu, "Structure-aware language model pretraining improves dense retrieval on structured data," *arXiv preprint arXiv:2305.19912*, 2023.

[77] M. Kang, J. M. Kwak, J. Baek, and S. J. Hwang, "Knowledge graph-augmented language models for knowledge-grounded dialogue generation," *arXiv preprint arXiv:2305.18846*, 2023.

[78] W. Shen, Y. Gao, C. Huang, F. Wan, X. Quan, and W. Bi, "Retrieval-generation alignment for end-to-end task-oriented dialogue system," *arXiv preprint arXiv:2310.08877*, 2023.

[79] T. Shi, L. Li, Z. Lin, T. Yang, X. Quan, and Q. Wang, "Dual-feedback knowledge retrieval for task-oriented dialogue systems," *arXiv preprint arXiv:2310.14528*, 2023.

[80] P. Ranade and A. Joshi, "Fabula: Intelligence report generation using retrieval-augmented narrative construction," *arXiv preprint arXiv:2310.13848*, 2023.

[81] X. Jiang, R. Zhang, Y. Xu, R. Qiu, Y. Fang, Z. Wang, J. Tang, H. Ding, X. Chu, J. Zhao *et al.*, "Think and retrieval: A hypothesis knowledge graph enhanced medical large language models," *arXiv preprint arXiv:2312.15883*, 2023.

[82] J. Baek, S. Jeong, M. Kang, J. C. Park, and S. J. Hwang, "Knowledge-augmented language model verification," *arXiv preprint arXiv:2310.12836*, 2023.

[83] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, "Reasoning on graphs: Faithful and interpretable large language model reasoning," *arXiv preprint arXiv:2310.01061*, 2023.

[84] X. He, Y. Tian, Y. Sun, N. V. Chawla, T. Laurent, Y. LeCun, X. Bresson, and B. Hooi, "G-retriever: Retrieval-augmented generation for textual graph understanding and question answering," *arXiv preprint arXiv:2402.07630*, 2024.

[85] L. Zha, J. Zhou, L. Li, R. Wang, Q. Huang, S. Yang, J. Yuan, C. Su, X. Li, A. Su *et al.*, "Tablegpt: Towards unifying tables, nature language and commands into one gpt," *arXiv preprint arXiv:2307.08674*, 2023.

[86] M. Gaur, K. Gunaratna, V. Srinivasan, and H. Jin, "Iseeq: Information seeking question generation using dynamic meta-information retrieval and knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 10 672–10 680.

[87] F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. H. Chi, N. Schärli, and D. Zhou, "Large language models can be easily distracted by irrelevant context," in *International Conference on Machine Learning*. PMLR, 2023, pp. 31 210–31 227.

[88] R. Teja, "Evaluating the ideal chunk size for a rag system using llamaindex," https://www.llamaindex.ai/blog/evaluating-the-ideal-chunk-size-for-a-rag-system-using-llamaindex-6207e5d3fec5, 2023.

[89] Langchain, "Recursively split by character," https://python.langchain.com/docs/modules/data_connection/document_transformers/recursive_text_splitter, 2023.

[90] S. Yang, "Advanced rag 01: Small-to-big retrieval," https://towardsdatascience.com/advanced-rag-01-small-to-big-retrieval-172181b396d4, 2023.

[91] Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr, "Knowledge graph prompting for multi-document question answering," *arXiv preprint arXiv:2308.11730*, 2023.

[92] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le *et al.*, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.

[93] S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz, and J. Weston, "Chain-of-verification reduces hallucination in large language models," *arXiv preprint arXiv:2309.11495*, 2023.

[94] X. Li and J. Li, "Angle-optimized text embeddings," *arXiv preprint arXiv:2309.12871*, 2023.

[95] VoyageAI, "Voyage's embedding models," https://docs.voyageai.com/embeddings/, 2023.

[96] BAAI, "Flagembedding," https://github.com/FlagOpen/FlagEmbedding, 2023.

[97] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J.-Y. Nie, "Retrieve anything to augment large language models," *arXiv preprint arXiv:2310.07554*, 2023.

[98] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, "Lost in the middle: How language models use long contexts," *arXiv preprint arXiv:2307.03172*, 2023.

[99] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, "Chat-rec: Towards interactive and explainable llms-augmented recommender system," *arXiv preprint arXiv:2303.14524*, 2023.

[100] N. Anderson, C. Wilson, and S. D. Richardson, "Lingua: Addressing scenarios for live interpretation and automatic dubbing," in *Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track)*, J. Campbell, S. Larocca, J. Marciano, K. Savenkov, and A. Yanishevsky, Eds. Orlando, USA: Association for Machine Translation in the Americas, Sep. 2022, pp. 202–209. [Online]. Available: https://aclanthology.org/2022.amta-upg.14

[101] H. Jiang, Q. Wu, X. Luo, D. Li, C.-Y. Lin, Y. Yang, and L. Qiu, "Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression," *arXiv preprint arXiv:2310.06839*, 2023.

[102] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[103] Y. Ma, Y. Cao, Y. Hong, and A. Sun, "Large language model is not a good few-shot information extractor, but a good reranker for hard samples!" *ArXiv*, vol. abs/2303.08559, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257532405

[104] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "Chatlaw: Open-source legal large language model with integrated external knowledge bases," *arXiv preprint arXiv:2306.16092*, 2023.

[105] O. Yoran, T. Wolfson, O. Ram, and J. Berant, "Making retrieval-augmented language models robust to irrelevant context," *arXiv preprint arXiv:2310.01558*, 2023.

[106] X. Li, R. Zhao, Y. K. Chia, B. Ding, L. Bing, S. Joty, and S. Poria, "Chain of knowledge: A framework for grounding large language models with structured knowledge bases," *arXiv preprint arXiv:2305.13269*, 2023.

[107] H. Yang, S. Yue, and Y. He, "Auto-gpt for online decision making: Benchmarks and additional opinions," *arXiv preprint arXiv:2306.02224*, 2023.

[108] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[109] J. Zhang, "Graph-toolformer: To empower llms with graph reasoning ability via prompt augmented by chatgpt," *arXiv preprint arXiv:2304.11116*, 2023.

[110] 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 arXiv:2112.09332*, 2021.

[111] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee *et al.*, "Natural questions: a benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[112] Y. Liu, S. Yavuz, R. Meng, M. Moorthy, S. Joty, C. Xiong, and Y. Zhou, "Exploring the integration strategies of retriever and large language models," *arXiv preprint arXiv:2308.12574*, 2023.

[113] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," *arXiv preprint arXiv:1705.03551*, 2017.

[114] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[115] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1533–1544.

[116] A. Mallen, A. Asai, V. Zhong, R. Das, H. Hajishirzi, and D. Khashabi, "When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories," *arXiv preprint arXiv:2212.10511*, 2022.

[117] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "Ms marco: A human-generated machine reading comprehension dataset," 2016.

[118] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *arXiv preprint arXiv:1809.09600*, 2018.

[119] X. Ho, A.-K. D. Nguyen, S. Sugawara, and A. Aizawa, "Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps," *arXiv preprint arXiv:2011.01060*, 2020.

[120] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Musique: Multihop questions via single-hop question composition," *Transactions of the Association for Computational Linguistics*, vol. 10, pp. 539–554, 2022.

[121] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli, "Eli5: Long form question answering," *arXiv preprint arXiv:1907.09190*, 2019.

[122] T. Kočiskỳ, J. Schwarz, P. Blunsom, C. Dyer, K. M. Hermann, G. Melis, and E. Grefenstette, "The narrativeqa reading comprehension challenge," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 317–328, 2018.

[123] K.-H. Lee, X. Chen, H. Furuta, J. Canny, and I. Fischer, "A human-inspired reading agent with gist memory of very long contexts," *arXiv preprint arXiv:2402.09727*, 2024.

[124] I. Stelmakh, Y. Luan, B. Dhingra, and M.-W. Chang, "Asqa: Factoid questions meet long-form answers," *arXiv preprint arXiv:2204.06092*, 2022.

[125] M. Zhong, D. Yin, T. Yu, A. Zaidi, M. Mutuma, R. Jha, A. H. Awadallah, A. Celikyilmaz, Y. Liu, X. Qiu *et al.*, "Qmsum: A new benchmark for query-based multi-domain meeting summarization," *arXiv preprint arXiv:2104.05938*, 2021.

[126] P. Dasigi, K. Lo, I. Beltagy, A. Cohan, N. A. Smith, and M. Gardner, "A dataset of information-seeking questions and answers anchored in research papers," *arXiv preprint arXiv:2105.03011*, 2021.

[127] T. Möller, A. Reina, R. Jayakumar, and M. Pietsch, "Covid-qa: A question answering dataset for covid-19," in *ACL 2020 Workshop on Natural Language Processing for COVID-19 (NLP-COVID)*, 2020.

[128] X. Wang, G. H. Chen, D. Song, Z. Zhang, Z. Chen, Q. Xiao, F. Jiang, J. Li, X. Wan, B. Wang *et al.*, "Cmb: A comprehensive medical benchmark in chinese," *arXiv preprint arXiv:2308.08833*, 2023.

[129] H. Zeng, "Measuring massive multitask chinese understanding," *arXiv preprint arXiv:2304.12986*, 2023.

[130] R. Y. Pang, A. Parrish, N. Joshi, N. Nangia, J. Phang, A. Chen, V. Padmakumar, J. Ma, J. Thompson, H. He *et al.*, "Quality: Question answering with long input texts, yes!" *arXiv preprint arXiv:2112.08608*, 2021.

[131] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord, "Think you have solved question answering? try arc, the ai2 reasoning challenge," *arXiv preprint arXiv:1803.05457*, 2018.

[132] A. Talmor, J. Herzig, N. Lourie, and J. Berant, "Commonsenseqa: A question answering challenge targeting commonsense knowledge," *arXiv preprint arXiv:1811.00937*, 2018.

[133] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston, "Wizard of wikipedia: Knowledge-powered conversational agents," *arXiv preprint arXiv:1811.01241*, 2018.

[134] H. Wang, M. Hu, Y. Deng, R. Wang, F. Mi, W. Wang, Y. Wang, W.-C. Kwan, I. King, and K.-F. Wong, "Large language models as source

planner for personalized knowledge-grounded dialogue," *arXiv preprint arXiv:2310.08840*, 2023.
[135] ——, "Large language models as source planner for personalized knowledge-grounded dialogue," *arXiv preprint arXiv:2310.08840*, 2023.
[136] X. Xu, Z. Gou, W. Wu, Z.-Y. Niu, H. Wu, H. Wang, and S. Wang, "Long time no see! open-domain conversation with long-term persona memory," *arXiv preprint arXiv:2203.05797*, 2022.
[137] T.-H. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, D. Vandyke, and S. Young, "Conditional generation and snapshot learning in neural dialogue systems," *arXiv preprint arXiv:1606.03352*, 2016.
[138] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.
[139] S. Li, H. Ji, and J. Han, "Document-level event argument extraction by conditional generation," *arXiv preprint arXiv:2104.05919*, 2021.
[140] S. Ebner, P. Xia, R. Culkin, K. Rawlins, and B. Van Durme, "Multi-sentence argument linking," *arXiv preprint arXiv:1911.03766*, 2019.
[141] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, and E. Simperl, "T-rex: A large scale alignment of natural language with knowledge base triples," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.
[142] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," *arXiv preprint arXiv:1706.04115*, 2017.
[143] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, "Hellaswag: Can a machine really finish your sentence?" *arXiv preprint arXiv:1905.07830*, 2019.
[144] S. Kim, S. J. Joo, D. Kim, J. Jang, S. Ye, J. Shin, and M. Seo, "The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning," *arXiv preprint arXiv:2305.14045*, 2023.
[145] A. Saha, V. Pahuja, M. Khapra, K. Sankaranarayanan, and S. Chandar, "Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.
[146] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2020.
[147] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.
[148] 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," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 346–361, 2021.
[149] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: a large-scale dataset for fact extraction and verification," *arXiv preprint arXiv:1803.05355*, 2018.
[150] N. Kotonya and F. Toni, "Explainable automated fact-checking for public health claims," *arXiv preprint arXiv:2010.09926*, 2020.
[151] R. Lebret, D. Grangier, and M. Auli, "Neural text generation from structured data with application to the biography domain," *arXiv preprint arXiv:1603.07771*, 2016.
[152] H. Hayashi, P. Budania, P. Wang, C. Ackerson, R. Neervannan, and G. Neubig, "Wikiasp: A dataset for multi-domain aspect-based summarization," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 211–225, 2021.
[153] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," *arXiv preprint arXiv:1808.08745*, 2018.
[154] S. Saha, J. A. Junaed, M. Saleki, A. S. Sharma, M. R. Rifat, M. Rahouti, S. I. Ahmed, N. Mohammed, and M. R. Amin, "Vio-lens: A novel dataset of annotated social network posts leading to different forms of communal violence and its evaluation," in *Proceedings of the First Workshop on Bangla Language Processing (BLP-2023)*, 2023, pp. 72–84.
[155] X. Li and D. Roth, "Learning question classifiers," in *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.
[156] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.
[157] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt, "Codesearchnet challenge: Evaluating the state of semantic code search," *arXiv preprint arXiv:1909.09436*, 2019.
[158] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano *et al.*, "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.
[159] R. Steinberger, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufis, and D. Varga, "The jrc-acquis: A multilingual aligned parallel corpus with 20+ languages," *arXiv preprint cs/0609058*, 2006.
[160] Y. Hoshi, D. Miyashita, Y. Ng, K. Tatsuno, Y. Morioka, O. Torii, and J. Deguchi, "Ralle: A framework for developing and evaluating retrieval-augmented large language models," *arXiv preprint arXiv:2308.10633*, 2023.
[161] J. Liu, "Building production-ready rag applications," https://www.ai.engineer/summit/schedule/building-production-ready-rag-applications, 2023.
[162] I. Nguyen, "Evaluating rag part i: How to evaluate document retrieval," https://www.deepset.ai/blog/rag-evaluation-retrieval, 2023.
[163] Q. Leng, K. Uhlenhuth, and A. Polyzotis, "Best practices for llm evaluation of rag applications," https://www.databricks.com/blog/LLM-auto-eval-best-practices-RAG, 2023.
[164] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," *arXiv preprint arXiv:2309.15217*, 2023.
[165] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, "Ares: An automated evaluation framework for retrieval-augmented generation systems," *arXiv preprint arXiv:2311.09476*, 2023.
[166] C. Jarvis and J. Allard, "A survey of techniques for maximizing llm performance," https://community.openai.com/t/openai-dev-day-2023-breakout-sessions/505213#a-survey-of-techniques-for-maximizing-llm-performance-2, 2023.
[167] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," *arXiv preprint arXiv:2309.01431*, 2023.
[168] Y. Liu, L. Huang, S. Li, S. Chen, H. Zhou, F. Meng, J. Zhou, and X. Sun, "Recall: A benchmark for llms robustness against external counterfactual knowledge," *arXiv preprint arXiv:2311.08147*, 2023.
[169] Y. Lyu, Z. Li, S. Niu, F. Xiong, B. Tang, W. Wang, H. Wu, H. Liu, T. Xu, and E. Chen, "Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models," *arXiv preprint arXiv:2401.17043*, 2024.
[170] P. Xu, W. Ping, X. Wu, L. McAfee, C. Zhu, Z. Liu, S. Subramanian, E. Bakhturina, M. Shoeybi, and B. Catanzaro, "Retrieval meets long context large language models," *arXiv preprint arXiv:2310.03025*, 2023.
[171] C. Packer, V. Fang, S. G. Patil, K. Lin, S. Wooders, and J. E. Gonzalez, "Memgpt: Towards llms as operating systems," *arXiv preprint arXiv:2310.08560*, 2023.
[172] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis, "Efficient streaming language models with attention sinks," *arXiv preprint arXiv:2309.17453*, 2023.
[173] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "Raft: Adapting language model to domain specific rag," *arXiv preprint arXiv:2403.10131*, 2024.
[174] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.
[175] U. Alon, F. Xu, J. He, S. Sengupta, D. Roth, and G. Neubig, "Neuro-symbolic language modeling with automaton-augmented retrieval," in *International Conference on Machine Learning*. PMLR, 2022, pp. 468–485.
[176] 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," *arXiv preprint arXiv:2211.12561*, 2022.
[177] J. Li, D. Li, S. Savarese, and S. Hoi, "Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," *arXiv preprint arXiv:2301.12597*, 2023.
[178] W. Zhu, A. Yan, Y. Lu, W. Xu, X. E. Wang, M. Eckstein, and W. Y. Wang, "Visualize before you write: Imagination-guided open-ended text generation," *arXiv preprint arXiv:2210.03765*, 2022.
[179] J. Zhao, G. Haffar, and E. Shareghi, "Generating synthetic speech from spokenvocab for speech translation," *arXiv preprint arXiv:2210.08174*, 2022.
[180] D. M. Chan, S. Ghosh, A. Rastrow, and B. Hoffmeister, "Using external off-policy speech-to-text mappings in contextual end-to-end automated speech recognition," *arXiv preprint arXiv:2301.02736*, 2023.

[181] A. Yang, A. Nagrani, P. H. Seo, A. Miech, J. Pont-Tuset, I. Laptev, J. Sivic, and C. Schmid, "Vid2seq: Large-scale pretraining of a visual language model for dense video captioning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 714–10 726.

[182] N. Nashid, M. Sintaha, and A. Mesbah, "Retrieval-based prompt selection for code-related few-shot learning," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, 2023, pp. 2450–2462.

Causal Inference — What If

5 min read · Jun 23, 2020

Ofir Shalev (@ofirdi) Follow

Judea Pearl, a pioneering figure in artificial intelligence, argues that AI has been stuck in a decades-long rut. His prescription for progress? Teach machines to understand the question why.

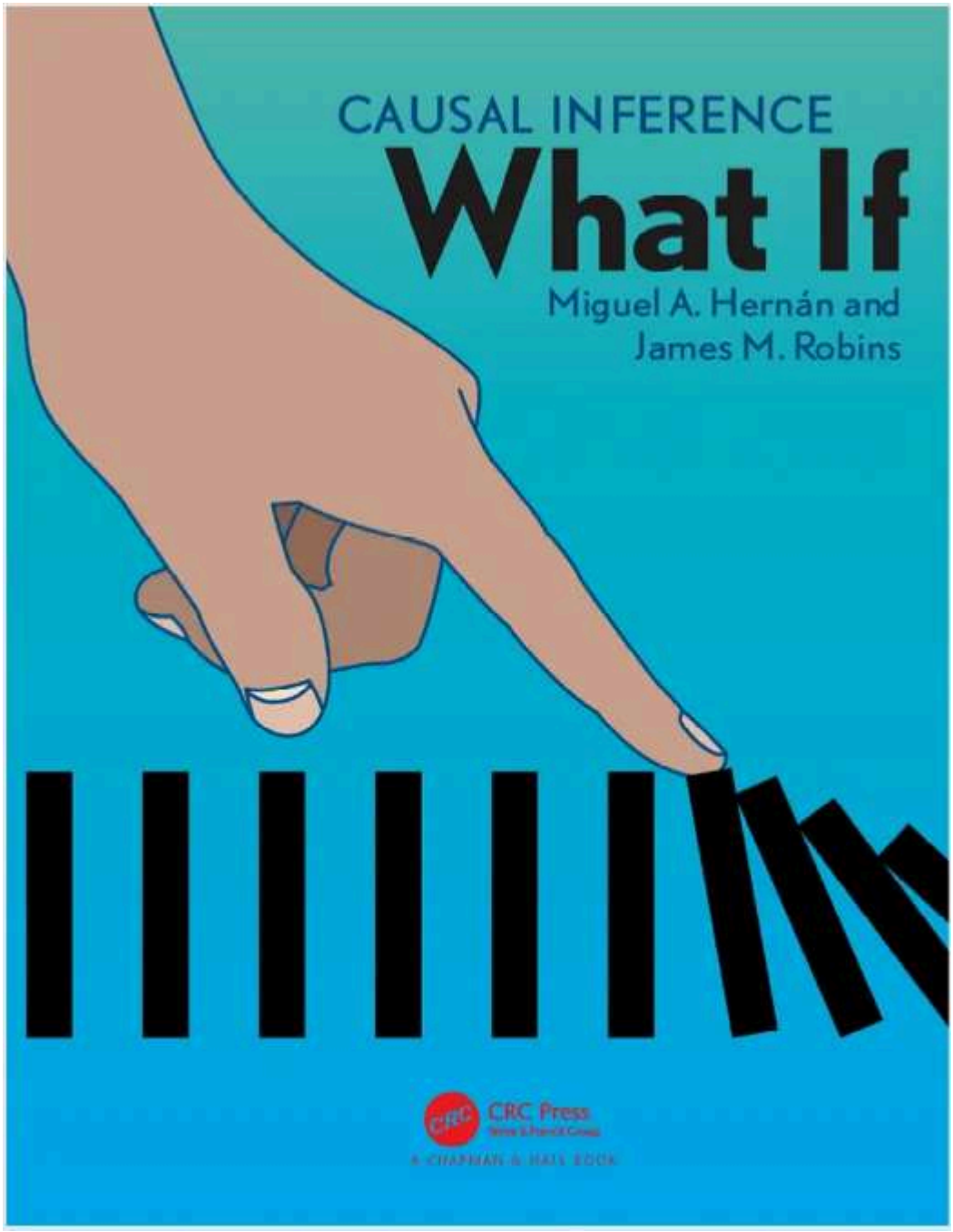
> All the impressive achievements of deep learning amount to just curve fitting
>
> Judea Pearl

Yoshua Bengio added in a recent interview

> Now, Bengio says deep learning needs to be fixed. He believes it won't realize its full potential, and won't deliver a true AI revolution, until it can go beyond pattern recognition and learn more about cause and effect. In other words, he says, deep learning needs to start asking *why* things happen.
>
> https://www.wired.com/story/ai-pioneer-algorithms-understand-why/

When we look at observational metrics, our Machine Learning models are doing great predicting a certain outcome given a treatment, but they are good exactly at that and not at the counterfactual — what would have been the outcome given no treatment

Causal Inference by Miguel Hernán and James Robins provides a great introduction to causal inference. You can download latest draft from their website:



Hernán MA, Robins JM (2020). Causal Inference: What If. Boca Raton: Chapman & Hall/CRC

The book is divided in three parts of increasing difficulty: **Part I** is about causal inference *without models* (i.e., nonparametric identification of causal effects), **Part II** is about causal inference with *models* (i.e., estimation of causal effects with parametric models), and **Part III** is about causal inference from *complex longitudinal data* (i.e., estimation of causal effects of time-varying treatments).

Here are the top *four* reasons of why I think it's a great book:

Detailed introduction to the key concepts including many examples

The first four chapters (a definition of causal effect, randomised experiments, observational studies and effect modification) cover key concepts such as potential outcomes (the outcome variable that *would have been observed* under a certain treatment value), individual and average causal effects, randomisation, identifiability conditions, exchangeability, positivity and consistency. You will get to know *Zeus's extended family,* with many examples covering their various health conditions and treatment options. As an example, table 1.1 shows the counterfactual outcomes (die or not) under both treatment (a = 1 a heart transplant) and no treatment (a = 0). Providing practical examples along with the definition helps cement the learning by identifying the key attributes associated with the concept.

Table 1.1

	$Y^{a=0}$	$Y^{a=1}$
Rheia	0	1
Kronos	1	0
Demeter	0	0
Hades	0	0
Hestia	0	0
Poseidon	1	0
Hera	0	0
Zeus	0	1
Artemis	1	1
Apollo	1	0
Leto	0	1
Ares	1	1
Athena	1	1
Hephaestus	0	1
Aphrodite	0	1
Cyclope	0	1
Persephone	1	1
Hermes	1	0
Hebe	1	0
Dionysus	1	0

Practical approach

Starting from the introduction, the authors are quite clear about their goals

> Importantly, this is not a philosophy book. We remain agnostic about metaphysical concepts like causality and cause. Rather, we focus on the identification and estimation of causal effects in populations, that is, numerical quantities that measure changes in the distribution of an outcome under different interventions. For example, we discuss how to estimate
> in patients with serious heart failure if they received a heart transplant versus if

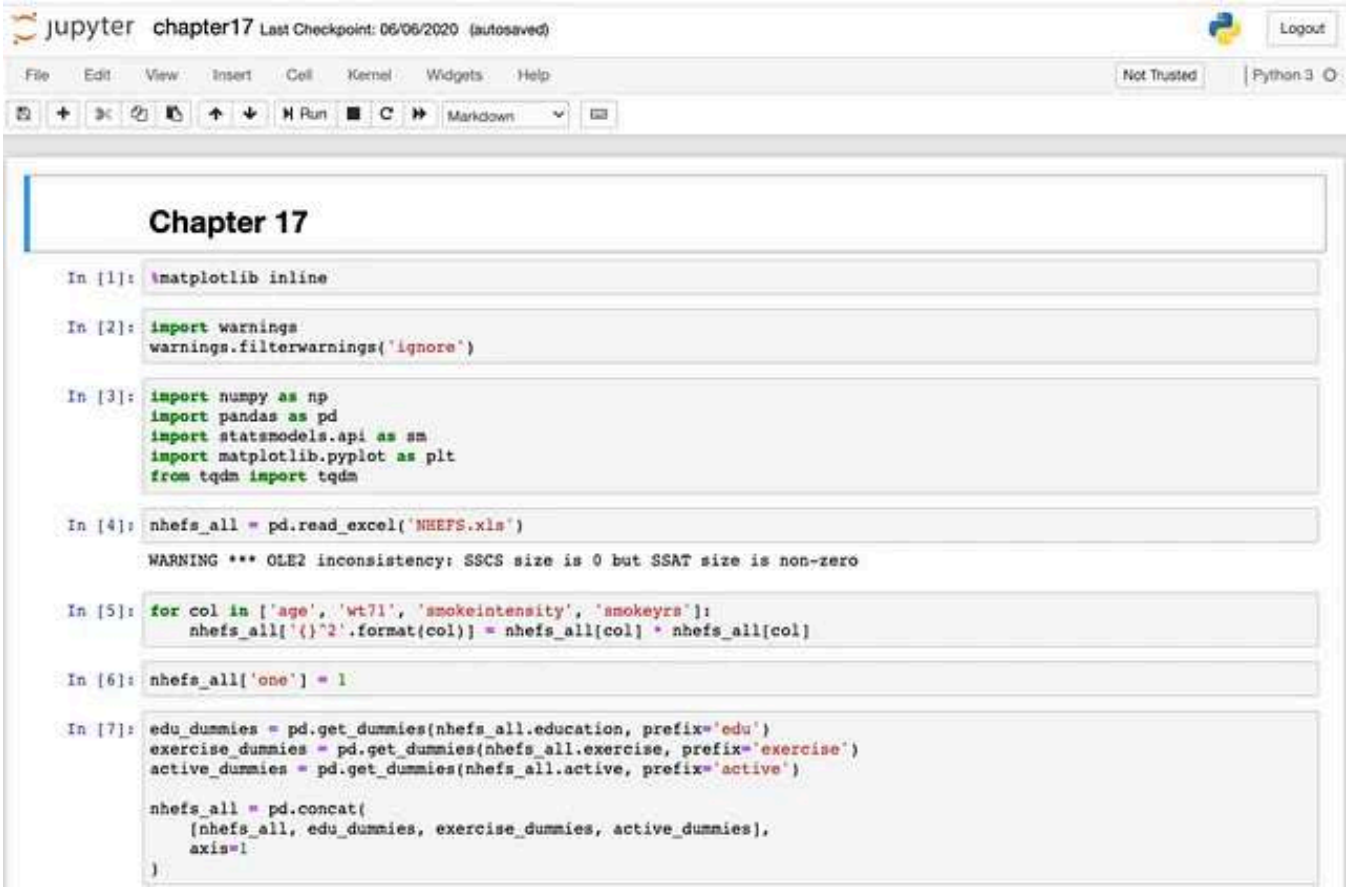
> they did not receive a heart transplant. Our main goal is to help decision makers make better decisions
>
> *Introduction: Towards less causal inferences*

On top of it, the book comes with a large number of code example in both R and Python, covering the first two part including chapters 11–17. It would be great to see additional code examples covering part three (causal inference from complex longitudinal data).

You should start with reading the book, and on parallel fire-up

```
jupyter notebook
```

and start playing with the code



The validity of causal inferences models

The authors discuss a large number of non-parametric and parametric techniques and algorithms to calculate causal effects. But they keep reminding us that all of these techniques rely on untestable assumptions and on expert knowledge. As an example:

> Unfortunately, no matter how many variables are included in L, there is no way to test that the assumption (conditional exchangeability) is correct, which makes causal inference from observational data a risky task. The validity of causal inferences requires that the investigators' expert knowledge is correct

and

> Causal inference generally requires expert knowledge and untestable assumptions about the causal network linking treatment, outcome, and other variables.

A (geeky) sense of humor

Technical books tend to be concise and dry, telling an anecdote or adding a joke can make difficult content more enjoyable and understandable.

As an example, when discussing the potential outcomes of the heart transplant treatment in Zeus's extended family, here is how the authors introduced the issue of

sampling variability:

> At this point you could complain that our procedure to compute effect measures is somewhat implausible. **Not only did we ignore the well known fact that the immortal Zeus cannot die**, but more to the point — our population in Table 1.1 had only 20 individuals.
>
> Chapter 1.4

As another example, chapter 7 introduces the topic of confounding variables using an observational study which is designed to answer the causal question “does one’s looking up to the sky make other pedestrians look up too?”. The plot develops and new details are being shared in chapters 8 (selection bias), chapter 9 (measurement bias) and chapter 10 (random variability), till the authors announce the following

> Do not worry. No more chapter introductions around the effect of your looking up on other people’s looking up. We squeezed that example well beyond what seemed possible
>
> Chapter 11

I hope that you will find this book useful and that you will enjoy learning about Causal Inference as much as I did!

** originally published here*

Machine Learning | Causal Inference | Python

Follow

Written by Ofir Shalev (@ofirdi)

155 followers · 3 following

* Chief Data Officer 📚 @GoJek (GoPay) http://bit.ly/LinkedInOfir http://bit.ly/technofob

How Gen AI is evolving: From mere Content Retrieval to Causal Reasoning

Introduction

Ever since ChatGPT was released in late 2022, Generative AI (Gen AI) has seen frenetic adoption in every sector. Yet, despite its remarkable ability to generate text, summarize,and assist in programming, underlying issues like hallucination and lack of reasoning have kept it from being utilized in high-consequence fields such as healthcare, finance, and scientific research. This whitepaper talks about the evolution of Gen AI, the limitations of current models, and the role of causal reasoning—more specifically Judea Pearl's work and causal inference—in overcoming these.

1. The Advent of Generative AI

The Breakthrough: ChatGPT and Large Language Models

OpenAI's ChatGPT led the way in mass Gen AI adoption by exhibiting almost human-level conversational capabilities. The model, developed based on transformer-based architectures, showed remarkable skills in:

- Content generation (e.g., text, code, creative writing)
- Text summarization and information retrieval
- Customer service conversational AI

Others, such as Google (Gemini), Anthropic (Claude), and Meta (LLaMA), followed shortly afterward, pushing the limits of large-scale language models (LLMs). Organizations started integrating Gen AI into their processes, anticipating more efficiency and automation.

Major Challenges: Hallucinations and Reliability Issues

For all their dazzling performance, however, LLMs have one unavoidable failing: hallucination—the generation of factually incorrect or entirely made-up information. The reason is that LLMs are programmed to learn how to generate the next word by predicting statistical relationships rather than knowing meaning or truth.

This restriction prevents Gen AI from being properly applied in domains where precision is vital. Organizations quickly understood that LLMs are potent but lack the ability to reason and consistently fail in complicated decision-making situations.

2. Addressing Hallucinations: Reasoning, RLHF, and RAG

Improving Reasoning Capabilities

A major research focus has been improving LLMs' reasoning abilities. Current models operate at a **correlation-based level**, recognizing patterns rather than true causal relationships. Efforts to mitigate hallucinations include:

- **Chain-of-Thought (CoT) prompting:** Encouraging models to break down reasoning steps explicitly
- **Self-consistency decoding:** Generating multiple responses and selecting the most consistent one
- **Tree of Thoughts (ToT):** Structuring decision-making in a tree format for better logical deduction

Reinforcement Learning from Human Feedback (RLHF) & RLAIF

To improve model reliability, OpenAI and other AI developers have employed **Reinforcement Learning from Human Feedback (RLHF)**, where human evaluators fine-tune model responses. However, RLHF is expensive and does not scale well. More recently, **Reinforcement Learning from AI Feedback (RLAIF)** has been proposed to automate evaluation by using AI critics instead of humans.

Retrieval-Augmented Generation (RAG)

One promising approach is **Retrieval-Augmented Generation (RAG)**, where models access external databases instead of relying solely on pre-trained knowledge. This enables real-time information retrieval and mitigates hallucinations by grounding responses in factual data sources. Companies like Microsoft, OpenAI, and Cohere are actively implementing RAG in enterprise applications.

3. Judea Pearl and the Role of Causal Inference in AI

The Problem: Correlation vs. Causation

Judea Pearl, a pioneer in causal reasoning, argues that current AI systems are limited to correlation-based learning and lack true understanding. In his landmark work, he proposed a framework for **causal inference**, enabling AI to distinguish between correlation and causation.

The Three Levels of Causal Reasoning

Pearl defines three levels of causal inference:

1. **Association (Seeing):** Recognizing statistical relationships (e.g., "When it rains, the streets are wet").
2. **Intervention (Doing):** Understanding cause-effect relationships (e.g., "If I turn on a sprinkler, the street gets wet").
3. **Counterfactuals (Imagining):** Answering "what-if" scenarios (e.g., "Would the street still be wet if it had not rained?").

Current LLMs operate at the first level (association), whereas true reasoning and decision-making require intervention and counterfactual thinking.

Causal Graphs and AI

Pearl developed causal Bayesian networks, which represent cause-and-effect relationships as directed acyclic graphs (DAGs). This enables AI models to:

•Infer causal relationships instead of just detecting patterns
• Foresee the outcomes of interventions
• Use counterfactual thinking to model different results

4. The Future: Combining Generative AI with Causal Inference

Integrating Causal Models into AI Systems

The next direction of AI research is to integrate causal inference into LLMs. Some potential approaches are:

- **Causal-enhanced RAG:** Using causal knowledge graphs to refine retrieved information
- **Hybrid AI Models:** Combining statistical deep learning with structured causal reasoning models

- **Simulation-Based AI:** Training AI agents to experiment and learn cause-effect relationships rather than just memorizing patterns

Challenges to Adoption

While promising, several challenges remain:

- **Scalability:** Causal models are more computationally intensive than correlation-based models
- **Data Limitations:** Causal inference often requires structured experimental data, which is scarce compared to unstructured text data
- **Industry Adoption:** The AI industry has been dominated by correlation-based deep learning, making a shift toward causal reasoning a paradigm change

5. Conclusion

Generative AI has transformed industries, but its shortcomings—most notably hallucinations—highlight the need for greater reasoning capability. Judea Pearl's work in causal inference provides a guide for taking AI from pattern recognition to true comprehension. By integrating causal models into Gen AI systems, we can increase reliability, interpretability, and decision-making, paving the way for the next generation of trustworthy AI applications.

Reducing hallucination in structured outputs via Retrieval-Augmented Generation

Patrice Béchard
ServiceNow
patrice.bechard@servicenow.com

Orlando Marquez Ayala
ServiceNow
orlando.marquez@servicenow.com

Abstract

A current limitation of Generative AI (GenAI) is its propensity to hallucinate. While Large Language Models (LLM) have taken the world by storm, without eliminating or at least reducing hallucination, real-world GenAI systems will likely continue to face challenges in user adoption. In the process of deploying an enterprise application that produces workflows from natural language requirements, we devised a system leveraging Retrieval-Augmented Generation (RAG) to improve the quality of the structured output that represents such workflows. Thanks to our implementation of RAG, our proposed system significantly reduces hallucination and allows the generalization of our LLM to out-of-domain settings. In addition, we show that using a small, well-trained retriever can reduce the size of the accompanying LLM at no loss in performance, thereby making deployments of LLM-based systems less resource-intensive.

1 Introduction

With the advent of Large Language Models (LLMs), structured output tasks such as converting natural language to code or to SQL have become commercially viable. A similar application is translating a natural language requirement to a *workflow*, a series of steps along with logic elements specifying their relationships. These workflows encapsulate processes that are executed automatically upon certain conditions, thereby increasing employee productivity. While enterprise systems offer such functionality to automate repetitive work and standardize processes, the barrier to entry is high, as building workflows requires specialized knowledge. Generative AI (GenAI) can lower this barrier since novice users can specify in natural language what they want their workflows to execute.

However, as with any GenAI application, using LLMs naively can produce ***untrustworthy*** outputs.

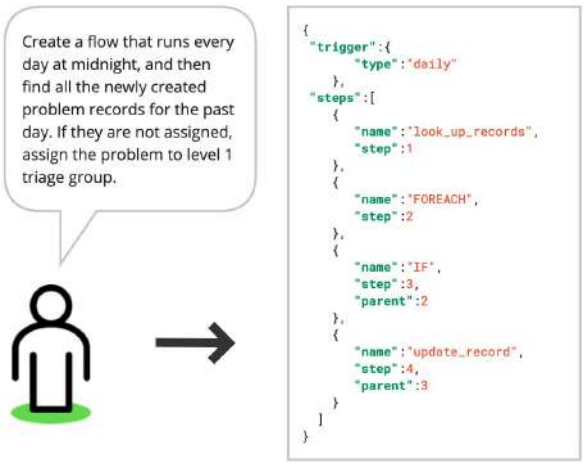


Figure 1: Sample structured output (JSON) to generate given a natural language requirement.

Such is the public concern for LLMs producing hallucinations that the Cambridge Dictionary chose *hallucinate* as its Word of the Year in 2023 (Cambridge, 2023). Retrieval-Augmented Generation (RAG) is a well-known method that can reduce hallucination and improve output quality, especially when generating the correct output requires access to external knowledge sources (Gao et al., 2024).

In this work, we describe how, in the process of building a commercial application that converts natural language to workflows, we employ RAG to improve the trustworthiness of the output by reducing hallucination. Workflows are represented as JSON documents where each step is a JSON object. Figure 1 shows an example of a text requirement and its associated JSON document. For simplicity, we include only the basic properties needed to identify a step along with properties indicating the relationship between steps. Besides the workflow steps, there may also be a *trigger* step that determines when the workflow should start, and sometimes this trigger requires a database table name. Hallucination in this task means generating properties such as steps or tables that do not exist.

While fine-tuning a sufficiently large LLM can

produce reasonably good workflows, the model may hallucinate, particularly if the natural language input is out-of-distribution. As the nature of enterprise users requires them to customize their applications, in this case by adding their own type of workflow steps, a commercial GenAI application needs to minimize the out-of-distribution mismatch. While one could fine-tune the LLM per enterprise, this may be prohibitively expensive due to the high infrastructure costs of fine-tuning LLMs. Another consideration when deploying LLMs is their footprint, making it preferable to deploy the smallest LLM that can perform the task.

Our contributions are the following:

- We provide an application of RAG in workflow generation, a structured output task.
- We show that using RAG reduces hallucination and improves results.
- We demonstrate that RAG allows deploying a smaller LLM while using a very small retriever model, at no loss in performance.

2 Related Work

Retrieval-Augmented Generation is a common approach to limit generation of false or outdated information in classical NLP tasks such as question answering and summarization (Lewis et al., 2020; Izacard and Grave, 2021; Shuster et al., 2021). In the GenAI era, it refers to a process where relevant information from specific data sources is retrieved prior to generating text; the generation is then based on this retrieved information (Gao et al., 2024). Our work differs from standard RAG as we apply it to a structured output task. Instead of retrieving facts, we retrieve JSON objects that could be part of the JSON output document. Providing plausible JSON objects to the LLM before generation increases the likelihood that the output JSON properties exist and that the generated JSON can be executed.

A crucial ingredient of RAG is the retriever since its output will be part of the LLM input. Compared to classical methods such as TF-IDF or BM25 that use lexical information, **Dense Retrieval** has been shown to be more effective as it maps the semantics to a multidimensional space where both queries and documents are represented (Reimers and Gurevych, 2019; Gao et al., 2021; Karpukhin et al., 2020; Xiong et al., 2020). These retrievers are often used in open-domain question answering systems (Guu et al., 2020; Lee et al., 2019), where both queries and documents are unstructured data and thus share the same semantic space. In our case, the queries are unstructured (natural language) and the documents (JSON objects) are structured. Our retrieval training is similar to Structure Aware DeNse ReTrievAl (SANTA), which proposes a training method to align the semantics between code and text (Li et al., 2023b).

Generating structured data falls within the realm of **Structured Output** tasks, which consist of generating a valid structured output from natural language, such as text-to-code, text-to-SQL (Zhong et al., 2017; Yu et al., 2018; Wang et al., 2020) or if-then program synthesis (Quirk et al., 2015; Liu et al., 2016; Dalal and Galbraith, 2020). They are challenging as they not only require generating output that can be parsed, but also entities or field values that exist in a given lexicon; otherwise the resulting output cannot be interpreted or compiled. For simple database schemas or small lexicons, this extra information can be included in the prompt. However, in our task the available pool of steps that can be part of a workflow is potentially very large and customizable per deployment, thereby making in-context learning impractical.

With the arrival of LLMs, these tasks have become more accessible. In particular, **Code LLMs** enable developers to write code faster by providing instructions to the LLM to generate code snippets (Chen et al., 2021; Nijkamp et al., 2022; Li et al., 2023a; Roziere et al., 2023). These models, trained on large datasets of source code (Kocetkov et al., 2022), have acquired broad knowledge of many programming languages and have been shown to perform better at tasks that necessitate reasoning (Madaan et al., 2022). Since the JSON schema to represent workflows is domain-specific, we cannot use these models off-the-shelf. While fine-tuning them on a small dataset increases the quality of results, extra steps are required to reduce hallucination and support out-of-domain queries.

Lastly, an alternative and complementary technique to reduce hallucination with LLMs is **Guided Generation** using tools such as Outlines (Willard and Louf, 2023). A sufficiently expressive context-free grammar could ensure that the steps generated by the model exist, but it does not provide extra knowledge as to which steps the flow should include given the natural language query.

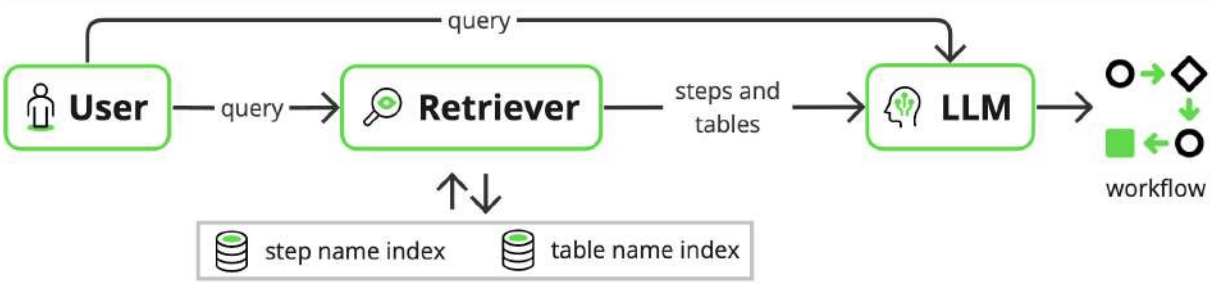


Figure 2: High-level architecture diagram showing how the user query is used by both the retriever and the LLM to generate the structured JSON output.

3 Methodology

Figure 2 depicts the high-level architecture of our RAG system. During initialization, indices of steps and tables are created using the retriever. When a user submits a request, the retriever is called to suggest steps and tables. The suggestions are then appended to the user query to form the LLM prompt. The LLM is then called to generate the workflow in the JSON format via greedy decoding.

To build our system, we first train a retriever encoder to align natural language with JSON objects. We then train an LLM in a RAG fashion by including the retriever's output in its prompt.

3.1 Retriever training

We expect the LLM to learn to construct JSON documents including the relationship between workflow steps, given sufficient examples. The risk of hallucination comes mainly from the step names since there are tens of thousands of possible steps and every customer can add their own steps if the default set does not meet their needs. In addition, as some trigger steps require database table names as a property, these names can also be hallucinated. We therefore require the retriever to map natural language to existing step and database table names.

We choose to fine-tune a retriever model for two reasons: to improve the mapping between text and JSON objects, and to create a better representation of the domain of our application. While there exist a myriad of open-source sentence encoders (Reimers and Gurevych, 2019; Ni et al., 2022), they have been trained in a setting where both queries and documents are in the same natural language semantic space. But in our case, the query or workflow requirement is unstructured while the JSON objects are structured data. Consistent with the results reported by Li et al. (2023b), who search code snippets based on text, fine-tuning improves the retrieval results greatly. Similarly, fine-tuning a model using our domain-specific data allows the retriever to learn the nuances and technicalities of the text and JSON that are particular to our setting.

We use a siamese transformer encoder with mean pooling similar to Reimers and Gurevych (2019) to encode both the user query and the step or table JSON object into fixed-length vectors. We include a normalization layer in our model so that the resulting embeddings have a norm of 1. We generate three embeddings $v_q \in \mathbb{R}^n$, $v_s \in \mathbb{R}^n$, $v_t \in \mathbb{R}^n$:

$$v_q = R(q) \qquad v_s = R(s) \qquad v_t = R(t) \quad (1)$$

where q, s, t are the user query, step, and table respectively. Retriever R can be decomposed as:

$$R(q) = \text{Norm}(\text{MeanPool}(\text{Enc}(q))) \quad (2)$$

The retriever model is trained on pairs of user queries and corresponding steps or tables. Since table names are used only in certain examples depending on the type of trigger, a query can be mapped to zero tables. For instance, the workflow in Figure 1 has four steps, forming four positive training pairs, each pair consisting of the same query and one of the steps in the flow. As the `daily` trigger step does not need a table name, the query is mapped to an empty list of tables.

We also construct negative training pairs by sampling steps or tables that are not relevant to the user query. We experiment with three different negative sampling strategies: random, BM25-based, and ANCE-based (Xiong et al., 2020).

The retriever is trained using a contrastive loss (Hadsell et al., 2006) to minimize the distance between positive pairs ($Y = 1$) and negative pairs ($Y = 0$). Given the cosine similarity between the query and step (or table) vectors, and cosine

distance $D = 1 - \text{cossim}(v_q, v_s)$, we define contrastive loss $\mathcal{L}$ as:

$$\mathcal{L} = \frac{1}{2}\left(YD^2 + (1-Y)\cdot\max(0, \frac{1}{2} - D)^2\right) \quad (3)$$

During initialization, we build an index of steps and tables using FAISS (Douze et al., 2024). When a user submits a natural language query, we embed the incoming query using our retriever and use cosine similarity to retrieve the max K steps and tables associated with this requirement.

3.2 LLM training

Contrary to end-to-end RAG systems such as Lewis et al. (2020), we opted to train both the retriever and LLM separately, for simplicity. We use the trained retriever to augment our dataset with suggested step and table names for each example. We then proceed with standard LLM supervised fine-tuning.

```
<|system|>
Tables:[{"name":"table","value":"issue"},
{"name":"table","value":"problem"}]
Steps:[{"name":"log"},
{"name":"update_record"},
{"name":"create_record"}]<|end|>
<|user|>
When a new issue arrives, log the time and date and
create a copy in the problem table<|end|>
<|assistant|>
{"trigger":{"type":"row_create",
"inputs":{"name":"table","value":"issue"}},
"steps":[{"name":"log","step":1},
{"name":"create_record","step":2}]}<|end|>
```

Figure 3: Training example, where the last four lines are the expected output (in red). The underlined text comes from the retriever's output.

By inserting the retriever's output in JSON format into the LLM input, we effectively make this structured output task easier as the LLM can copy the relevant JSON objects during generation. Figure 3 shows an example of a training example. Every line except the last four make up the LLM prompt. The suggested tables and steps come before the user query and are underlined in the figure. We exclude the most frequent steps from these suggestions as we expect the LLM to memorize them. Also, in every LLM training example, we assume the retriever has 100% recall: the steps and table required to build the structured output are always in the suggestions, except for the most frequent steps.

As we are showing the LLM thousands of examples during training, we did not find it necessary to experiment with complicated or verbose prompts: we used a short and simple format, similar to Figure 3, to reduce the number of input tokens while making it clear that this is a structured output task. As shown in section 5.2, this approach yielded good performance.

4 Experiments

As the task we are interested in is part of a commercial enterprise system, we had to devise our own datasets as well as evaluation metrics.

4.1 Datasets

From internal deployments of our enterprise platform, we extracted around 4,000 examples of deployed workflows and asked annotators to write natural language requirements for them. In addition, using deterministic rules, we created around 1,000 samples having simple and few steps in order to teach the model to handle input where the user is incrementally building their workflow. To have an unbiased estimate of the quality of results once the system is deployed, we asked expert users to simulate interacting with the system through a simple user interface where they typed their requirement. We used these interactions and the expected JSON documents to create an additional dataset split, named "Human Eval." Our final metrics are based on this split instead of the "Test" split, due to its higher quality and more realistic input. Table 1 shows statistics for all of our in-domain splits. Not all samples require triggers, and a small subset require the model to generate tables.

Split	Size	# Triggers	# Tables
Train	2867	823	556
Dev	318	77	44
Test	798	247	163
Human Eval	157	99	60

Table 1: Data statistics for in-domain training and evaluation.

A drawback of our data labeling approach is that these internal datasets are mostly in the IT domain, whereas our RAG system can be deployed in diverse domains such as HR and finance. Without assessing the quality of the system in out-of-distribution settings, we cannot be confident that the system will behave as expected. We therefore asked annotators to label five other splits, which come from other deployments of our enterprise platform. These are real workflows that have been created by real users.

Table 2 includes statistics for these out-of-domain splits. A measure of how different they are from our training data is the % of steps that are not in the set of steps in the "Train" split. This discrepancy ranges from less than 10% to more than 70%, highlighting the need to use a retriever and to customize the indices per deployment.

Split	Size	# Triggers	# Tables	% Steps not in Train
OOD1	146	133	47	49%
OOD2	162	111	21	76%
OOD3	429	226	114	34%
OOD4	42	25	11	33%
OOD5	353	271	26	7%

Table 2: Data statistics for out-of-domain evaluation.

To train the retriever encoder, we create pair examples out of the 4,000 extracted and 1,000 deterministically generated samples, resulting in around 15,000 pairs in the step names dataset and 1,500 in the table names dataset. The quality of this encoder is evaluated on the "Human Eval" split described above.

4.2 Metrics

We evaluate the entire RAG system using three metrics, which can all range from 0 to 1:

- **Trigger Exact Match (EM)** verifies whether the generated JSON trigger is exactly the same as the ground-truth, including the table name if this trigger requires it.
- **Bag of Steps (BofS)** measures the overlap between the generated JSON steps and the ground-truth steps in an order-agnostic fashion, akin to a bag-of-words approach.
- **Hallucinated Tables (HT)** and **Hallucinated Steps (HS)** measure the % of generated tables/steps that do not exist per workflow, indicating that they were invented by the LLM. This is the only metric where lower is better.

To evaluate the retriever, we use **Recall@15** for steps and **Recall@10** for tables. That is, given a natural language requirement, we retrieve the top K steps/tables from their respective indices and verify whether they cover the set of steps and the table, if required, included in the JSON document representing the workflow.

4.3 Models

As this is a production system, we have a trade-off between model size and performance for both the LLM and the retriever encoder.

We fine-tune models of different sizes to measure the impact of model size on the final metrics. As StarCoderBase (Li et al., 2023a) has been pretrained on JSON in addition to many programming languages and comes in different sizes, we fine-tune its 1B, 3B, 7B and 15.5B variants. Given our infrastructure constraints, we could deploy an LLM of at most 7B parameters. Thus we also fine-tune other pretrained LLMs of this size: CodeLlama-7B (Roziere et al., 2023) and Mistral-7B-v0.1 (Jiang et al., 2023). All the LLMs were fine-tuned using the same datasets and hyperparameters.

We use all-mpnet-base-v2[1] as the base retriever model. As it has only 110M parameters, it is suitable for deployment. We compare our fine-tuned model against different sizes of off-the-shelf GTR-T5 models (Ni et al., 2022) to see whether larger encoders impact the performance.

Please see Appendix A for training details for both the LLM and the retriever encoder.

5 Results

5.1 Retriever encoder

Table 3 shows the results of retrieval on the "Human Eval" split for both steps and tables. Scaling the size of the off-the-shelf encoders, as we did with GTR-T5, does not yield significant improvements on both retrieval metrics. A similar observation was made by Neelakantan et al. (2022) for code retrieval. What was crucial to significantly improve the performance was fine-tuning the encoder.

Model (# Params)	Step Recall@15	Table Recall@10
gtr-t5-base (110M)	0.505	0.489
gtr-t5-large (355M)	0.575	0.511
gtr-t5-xl (1.24B)	0.579	0.489
gtr-t5-xxl (4.8B)	0.561	0.489
all-mpnet-base-v2 (110M)	0.425	0.170
+ Random	0.640	0.752
+ BM25	0.537	0.586
+ ANCE	0.556	0.699
+ All	**0.743**	**0.766**

Table 3: Evaluation of different encoders on step and table retrieval. The last four rows represent encoders fine-tuned using different negative sampling strategies.

Due to deployment considerations, we fine-tune the smallest encoders (110M parameters), and found that all-mpnet-base-v2 yielded the best performance after fine-tuning with all negative sampling strategies.

[1] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Model	Trigger EM	Bag of Steps	Hallucinated Steps	Hallucinated Tables
No Retriever				
StarCoderBase-1B	0.580	0.645	0.157	0.192
StarCoderBase-3B	0.551	0.648	0.140	0.214
StarCoderBase-7B	0.547	**0.669**	0.137	0.206
StarCoderBase (15.5B)	0.632	0.662	0.160	0.194
With Retriever				
StarCoderBase-1B	0.591	0.619	0.072	0.044
StarCoderBase-3B	0.615	0.641	**0.017**	0.030
StarCoderBase-7B	**0.664**	**0.672**	**0.019**	0.042
StarCoderBase (15.5B)	**0.667**	**0.667**	0.040	**0.016**
CodeLlama-7B	0.623	0.617	0.039	0.108
Mistral-7B-v0.1	0.596	0.617	0.049	0.045

Table 4: Performance of various model types and sizes on the "Human Eval" split. Lower is better for the hallucination metrics. Results within 0.005 of the best score are highlighted in **bold**.

5.2 Retrieval-Augmented Generation

Our main objective is to reduce hallucination while keeping the overall performance high given our infrastructure constraints. Table 4 shows that without a retriever (only LLM fine-tuning), the % of hallucinated steps and tables can be as high as 21% on the "Human Eval" split. Using a retriever, this decreases to less than 7.5% for steps and less than 4.5% for tables with all StarCoderBase LLMs. All models produce valid JSON documents following the expected schema, thanks to fine-tuning.

Without a retriever, scaling the size of the StarCoderBase models improves the Bag of Steps and Trigger Exact Match metrics, albeit unevenly. Scaling also helps with RAG, but we observe more consistent improvements. This suggests that larger LLMs can better copy and paste retrieved steps and tables during generation.

The smallest RAG fine-tuned model (1B) hallucinates significantly more than its larger counterparts. Among the other three variants, the 7B version gives us the best trade-off, as the performance difference between 7B and 15.5B is marginal. Another observation is that the 3B version trained with RAG is competitive even with the 15.5B version without RAG on the Trigger EM and Bag of Steps metrics, while keeping hallucination low. This is a key lesson as we could deploy a 3B RAG fine-tuned model if we had more limited infrastructure.

Lastly, we compare the RAG fine-tuned StarCoderBase-7B to fine-tuning more recent LLMs of the same size. Despite also fine-tuning them with RAG, CodeLlama-7B and Mistral-7B-v0.1 produce worse results across all metrics, even compared to the smaller StarCoderBase-3B. We suspect that pre-training on large amounts of natural language data may be detrimental to our task.

5.3 OOD evaluation

We want our approach to perform well on OOD scenarios without further fine-tuning the retriever or the LLM. Table 5 assesses the performance of our chosen RAG fine-tuned StarCoderBase-7B model on the five OOD splits described by Table 2.

Split	Trigger EM	BofS	HS	HT
OOD1	0.662	0.619	0.063	0.051
OOD2	0.645	0.612	0.020	0.151
OOD3	0.562	0.743	0.014	0.033
OOD4	0.400	0.671	0.011	0.154
OOD5	0.774	0.770	0.005	0.063
Avg.	0.647	0.714	0.018	0.066
No RAG Avg.	0.544	0.629	0.020	0.428
Human Eval	0.664	0.672	0.019	0.042

Table 5: Performance of RAG fine-tuned StarCoderBase-7B on OOD splits.

We observe that on average, thanks to the retriever, all the OOD metrics are similar to the in-domain results represented by the "Human Eval" split. We use a weighted average based on the number of samples per split.

To quantify the effect of suggesting step and table names, we evaluate the RAG fine-tuned StarCoderBase-7B model without suggestions in row "No RAG Avg.". All metrics worsen significantly while the "Hallucinated Steps" remains roughly the same. Upon inspection, we see that the RAG fine-tuned model has learned to be conservative in generating steps when it does not receive suggestions, relying only on steps that it has seen during training. On the other hand, the "Hallucinated Tables" metric is significantly worse as the model is more creative when it comes to tables. Please see Appendix B for supplementary detail.

5.4 Error Analysis

When investigating error patterns found in the generated workflows, we observe issues arising from failures both on the retriever and the LLM.

For complex flows where steps that are used less frequently need to be retrieved, if a crucial component is not in the retriever's suggestions, it becomes difficult for the LLM to generate a valid workflow in line with the user query. To improve the retriever's recall, we can decompose the query into shorter texts to make the retrieval step more precise for each step. This would mean performing several retrieval calls, potentially one per step, instead of making one single retrieval call as we are doing now.

In some cases, the LLM did not produce the desired structure. This is more often seen when using steps that determine the logic of the workflow, such as `IF`, `TRY`, or `FOREACH`. These are important errors that can be addressed by synthetic data generation after analyzing which steps are being missed. For examples of perfect output and when the retriever and LLM fail, please refer to Appendix C.

5.5 Impact on Engineering

The obtained results led us to make several decisions that impacted the scalability and modularity of the system. Since the best overall performance was given by a 7B-parameter model, we could have a larger batch size for incoming user requests, thereby increasing the system throughput given a single GPU. This implies a trade-off in latency as larger queries (in number of tokens) result in larger number of generated tokens, sometimes causing large queries to become a bottleneck if they are included in a batch with many shorter queries. Our stress tests and user research reveal that the current system overall response time is acceptable.

Obtaining good results after fine-tuning a very small encoder for the retriever (110M parameters), allowed us to deploy it on the same GPU with negligible effect on the larger LLM. But we could even deploy the retriever on CPU due to its small size. A benefit of not performing joint training between the retriever and the LLM is that the retriever can be reused for other use cases involving similar data sources. Moreover, decoupling them allows clearer separation of concerns and independent optimization by separate team members. Nevertheless, for scientific purposes, it is still worthwhile to experiment with joint training.

We have several ideas to reduce the system response time: changing the structured output format from JSON to YAML to reduce the number of tokens, leveraging speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Joao Gante, 2023), and streaming one step at a time back to the user instead of the entire generated workflow.

6 Conclusion

We propose an approach to deploy a Retrieval-Augmented LLM to reduce hallucination and allow generalization in a structured output task. Reducing hallucination is a sine qua non for users to adopt real-world GenAI systems. We show that RAG allows deploying a system in limited-resource settings as a very small retriever can be coupled with a small LLM. Future work includes improving the synergy between the retriever and the LLM, through joint training or a model architecture that allows them to work better together.

Ethical Considerations

While our work proposes an approach to reduce hallucination in structure output tasks, we do not claim that the risk of harm due to hallucination is eliminated. Our deployed system includes a layer of post-processing to clearly indicate to users the generated steps that do not exist and urge them to fix the output before continuing their work.

Acknowledgements

We thank our ServiceNow colleagues who worked hard in building the aforementioned system, from project managers to quality engineers. We also thank the several colleagues who reviewed an earlier version of this paper: Lindsay Brin, Hessam Amini, Erfan Hosseini, and Gabrielle Gauthier-Melançon, as well as the NAACL reviewers, for their valuable feedback.

References

Cambridge. 2023. Why hallucinate? https://dictionary.cambridge.org/editorial/woty.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph,

Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Dhairya Dalal and Byron V Galbraith. 2020. Evaluating sequence-to-sequence learning models for if-then program synthesis. *arXiv preprint arXiv:2002.03485*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pages 6894–6910. Association for Computational Linguistics (ACL).

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3929–3938.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2006*, pages 1735–1742.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880. Association for Computational Linguistics.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Joao Gante. 2023. Assisted generation: a new direction toward low-latency text generation.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Denis Kocetkov, Raymond Li, LI Jia, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, et al. 2022. The stack: 3 tb of permissively licensed source code. *Transactions on Machine Learning Research*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 9459–9474.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023a. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Xinze Li, Zhenghao Liu, Chenyan Xiong, Shi Yu, Yu Gu, Zhiyuan Liu, and Ge Yu. 2023b. Structure-aware language model pretraining improves dense retrieval on structured data. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11560–11574, Toronto, Canada. Association for Computational Linguistics.

Chang Liu, Xinyun Chen, Eui Chul Shin, Mingcheng Chen, and Dawn Song. 2016. Latent attention for if-then program synthesis. *Advances in Neural Information Processing Systems*, 29.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005*.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9844–9855.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. In *The Eleventh International Conference on Learning Representations*.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for llms. *arXiv preprint arXiv:2307.09702*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Training details for LLM and retriever

All LLMs were fine-tuned using the same set of hyperparameters. We use the AdamW optimizer with a learning rate of $5e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay of 0.01. Models were trained for 5,000 steps with a cosine learning rate scheduler with 100 warmup steps. We use an effective batch size of 32 for all models, using gradient accumulation when the batch size would not fit on a single GPU. We trained all models using LoRA (Hu et al., 2021) with $r = 16$, $\alpha = 16$ and a dropout rate of 0.05. All models were trained with flash-attention (Dao et al., 2022) on a single A100 80GB GPU.

We fine-tuned the retriever model using the SentenceTransformers framework (Reimers and Gurevych, 2019). We use the AdamW optimizer (Loshchilov and Hutter, 2018) and a learning rate of $2e-5$. We use a batch size of 128 and train the model for 10 epochs.

B Differences in generation with and without suggestions

To understand the impact of suggesting step and table names during generation, for each OOD split, we inspect the % of unique steps and % of unique table names that are hallucinated with and without suggestions.

Table 6 shows that without suggestions, the RAG fine-tuned StarCoderBase-7B tends to generate significantly fewer unique step names. Receiving suggestions allows the model to copy the suggestions, thereby increasing the diversity of what it generates. In addition, without suggestions a greater percentage of the unique step names it generates are invented.

	No suggestions		**With suggestions**	
Split	# unique steps	% H	# unique steps	% H
OOD1	52	40%	100	13%
OOD2	38	34%	96	13%
OOD3	122	37%	269	9%
OOD4	20	5%	32	9%
OOD5	88	17%	151	3%

Table 6: Statistics of generated step names in terms of uniqueness and hallucination. H refers to unique hallucinated step names.

We also see that even with suggestions, there is still an important gap in the percentage of unique step names that are hallucinated, as in some splits more than 10% of unique steps are invented. While the overall hallucination rate is less than 2%, as shown in Table 5, there are cases where the retriever does not suggest what is expected or the LLM does not take into account the suggestions.

	No suggestions		**With suggestions**	
Split	# unique tables	% H	# unique tables	% H
OOD1	40	70%	22	14%
OOD2	31	71%	19	21%
OOD3	61	64%	44	9%
OOD4	11	54%	9	22%
OOD5	38	68%	29	17%

Table 7: Statistics of generated table names in terms of uniqueness and hallucination. H refers to unique hallucinated table names

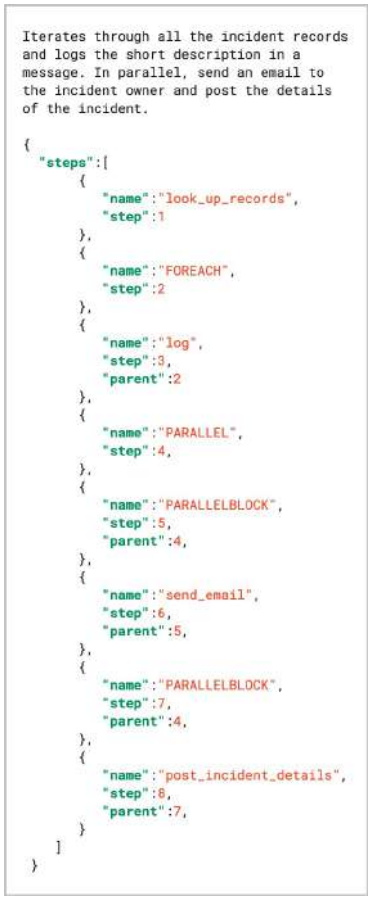
When it comes to table names, there are similar and different observations, as shown in Table 7. As in the case of step names, without suggestions a greater percentage of unique table names are invented. However, when provided with suggestions, the model is more conservative as it generates fewer unique table names. This may be an artifact of the data, where there is less diversity of tables used compared to step names.

C Sample perfect output and errors

Figure 4 shows three user queries along with their generated workflows. The first one is a complicated workflow where the LLM is able to follow exactly the structure described in the user query, and is able to use the steps that the user expected. In this case, the retriever suggests only the step `post_incident_details`, as the rest are considered common steps.

In the second example, the retriever fails to suggest the `send_slack_message` step. The resulting workflow is not entirely wrong but it is of lesser quality as the LLM uses the common step `send_notification`, which is not what the user expected.

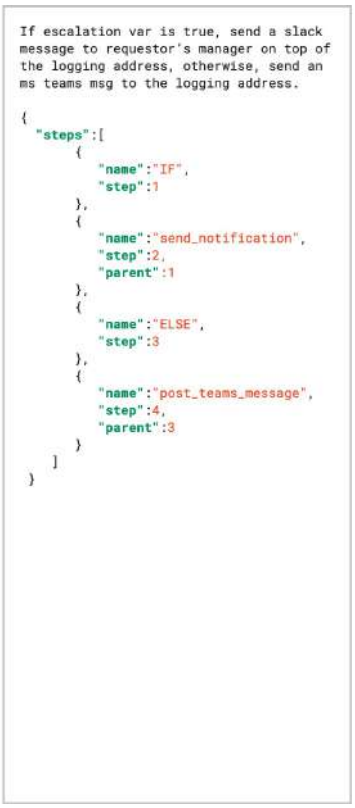
In the last example, the LLM shows that it does not sufficiently understand the semantics of the task. The word *Try* in the user query should have made it use the `TRY` and `CATCH` flow logic, but the LLM seems to ignore this word, resulting in a workflow that does not reflect what the user asked for.



```
Iterates through all the incident records
and logs the short description in a
message. In parallel, send an email to
the incident owner and post the details
of the incident.

{
  "steps":[
        {
           "name":"look_up_records",
           "step":1
        },
        {
           "name":"FOREACH",
           "step":2
        },
        {
           "name":"log",
           "step":3,
           "parent":2
        },
        {
           "name":"PARALLEL",
           "step":4,
        },
        {
           "name":"PARALLELBLOCK",
           "step":5,
           "parent":4,
        },
        {
           "name":"send_email",
           "step":6,
           "parent":5,
        },
        {
           "name":"PARALLELBLOCK",
           "step":7,
           "parent":4,
        },
        {
           "name":"post_incident_details",
           "step":8,
           "parent":7,
        }
    ]
}
```

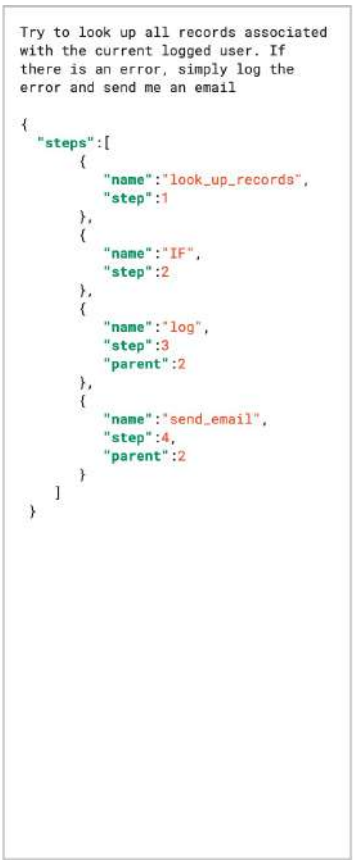
(a) Perfect output



```
If escalation var is true, send a slack
message to requestor's manager on top of
the logging address, otherwise, send an
ms teams msg to the logging address.

{
  "steps":[
        {
           "name":"IF",
           "step":1
        },
        {
           "name":"send_notification",
           "step":2,
           "parent":1
        },
        {
           "name":"ELSE",
           "step":3
        },
        {
           "name":"post_teams_message",
           "step":4,
           "parent":3
        }
    ]
}
```

(b) Retrieval error



```
Try to look up all records associated
with the current logged user. If
there is an error, simply log the
error and send me an email

{
  "steps":[
        {
           "name":"look_up_records",
           "step":1
        },
        {
           "name":"IF",
           "step":2
        },
        {
           "name":"log",
           "step":3
           "parent":2
        },
        {
           "name":"send_email",
           "step":4,
           "parent":2
        }
    ]
}
```

(c) LLM error

Figure 4: Examples where both the retriever and the LLM worked perfectly and where each of them failed: (a) All expected step names were suggested and used by the LLM. (b) The retriever did not suggest the step `send_slack_message` and therefore the LLM used the common step `send_notification` instead. (c) The LLM should have used the `TRY` step as the parent to all the steps, but it did not fully understand the user query.

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*]Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†]Work performed while at Google Brain.

[‡]Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

2 Background

The goal of reducing sequential computation also forms the foundation of the Extended Neural GPU [16], ByteNet [18] and ConvS2S [9], all of which use convolutional neural networks as basic building block, computing hidden representations in parallel for all input and output positions. In these models, the number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions, linearly for ConvS2S and logarithmically for ByteNet. This makes it more difficult to learn dependencies between distant positions [12]. In the Transformer this is reduced to a constant number of operations, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with Multi-Head Attention as described in section 3.2.

Self-attention, sometimes called intra-attention is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence. Self-attention has been used successfully in a variety of tasks including reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations [4, 27, 28, 22].

End-to-end memory networks are based on a recurrent attention mechanism instead of sequence-aligned recurrence and have been shown to perform well on simple-language question answering and language modeling tasks [34].

To the best of our knowledge, however, the Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution. In the following sections, we will describe the Transformer, motivate self-attention and discuss its advantages over models such as [17, 18] and [9].

3 Model Architecture

Most competitive neural sequence transduction models have an encoder-decoder structure [5, 2, 35]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [10], consuming the previously generated symbols as additional input when generating the next.

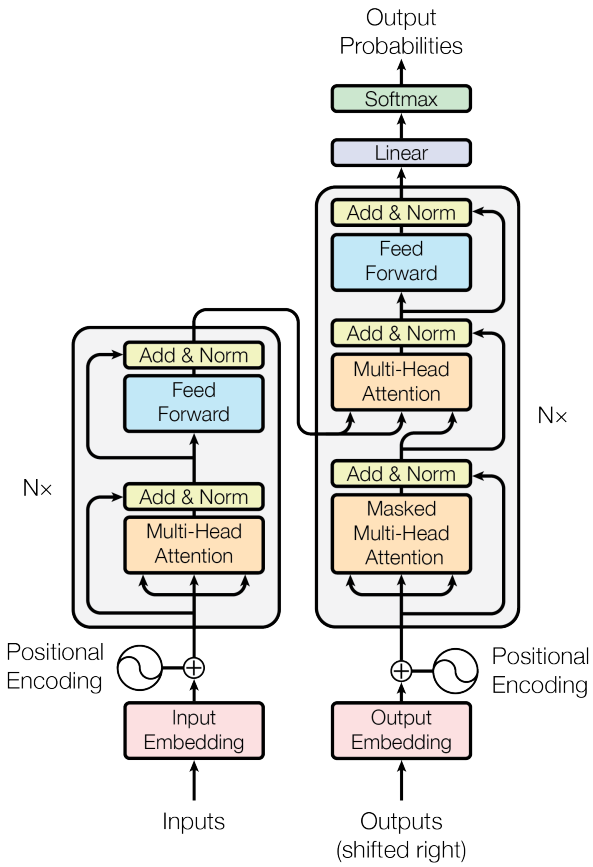


Figure 1: The Transformer - model architecture.

The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively.

3.1 Encoder and Decoder Stacks

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

3.2 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum

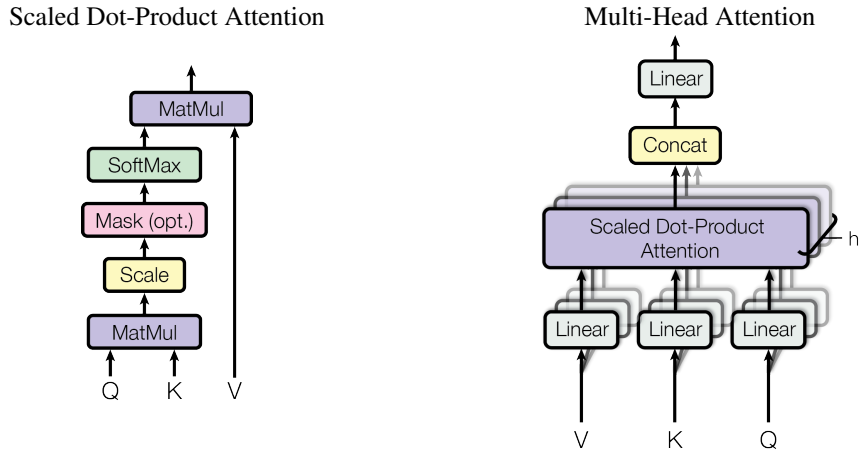


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.1 Scaled Dot-Product Attention

We call our particular attention "Scaled Dot-Product Attention" (Figure 2). The input consists of queries and keys of dimension d_k, and values of dimension d_v. We compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values.

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

The two most commonly used attention functions are additive attention [2], and dot-product (multiplicative) attention. Dot-product attention is identical to our algorithm, except for the scaling factor of $\frac{1}{\sqrt{d_k}}$. Additive attention computes the compatibility function using a feed-forward network with a single hidden layer. While the two are similar in theoretical complexity, dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

While for small values of d_k the two mechanisms perform similarly, additive attention outperforms dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k, the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

3.2.2 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions, respectively. On each of these projected versions of queries, keys and values we then perform the attention function in parallel, yielding d_v-dimensional

[4] To illustrate why the dot products get large, assume that the components of q and k are independent random variables with mean 0 and variance 1. Then their dot product, $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$, has mean 0 and variance d_k.

output values. These are concatenated and once again projected, resulting in the final values, as depicted in Figure 2.

Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.

In this work we employ $h = 8$ parallel attention layers, or heads. For each of these we use $d_k = d_v = d_{\text{model}}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

3.2.3 Applications of Attention in our Model

The Transformer uses multi-head attention in three different ways:

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models such as [38, 2, 9].
- The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connections. See Figure 2.

3.3 Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network, which is applied to each position separately and identically. This consists of two linear transformations with a ReLU activation in between.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2}$$

While the linear transformations are the same across different positions, they use different parameters from layer to layer. Another way of describing this is as two convolutions with kernel size 1. The dimensionality of input and output is $d_{\text{model}} = 512$, and the inner-layer has dimensionality $d_{ff} = 2048$.

3.4 Embeddings and Softmax

Similarly to other sequence transduction models, we use learned embeddings to convert the input tokens and output tokens to vectors of dimension d_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In our model, we share the same weight matrix between the two embedding layers and the pre-softmax linear transformation, similar to [30]. In the embedding layers, we multiply those weights by $\sqrt{d_{\text{model}}}$.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

4 Why Self-Attention

In this section we compare various aspects of self-attention layers to the recurrent and convolutional layers commonly used for mapping one variable-length sequence of symbol representations $(x_1, ..., x_n)$ to another sequence of equal length $(z_1, ..., z_n)$, with $x_i, z_i \in \mathbb{R}^d$, such as a hidden layer in a typical sequence transduction encoder or decoder. Motivating our use of self-attention we consider three desiderata.

One is the total computational complexity per layer. Another is the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required.

The third is the path length between long-range dependencies in the network. Learning long-range dependencies is a key challenge in many sequence transduction tasks. One key factor affecting the ability to learn such dependencies is the length of the paths forward and backward signals have to traverse in the network. The shorter these paths between any combination of positions in the input and output sequences, the easier it is to learn long-range dependencies [12]. Hence we also compare the maximum path length between any two input and output positions in networks composed of the different layer types.

As noted in Table 1, a self-attention layer connects all positions with a constant number of sequentially executed operations, whereas a recurrent layer requires $O(n)$ sequential operations. In terms of computational complexity, self-attention layers are faster than recurrent layers when the sequence

length n is smaller than the representation dimensionality d, which is most often the case with sentence representations used by state-of-the-art models in machine translations, such as word-piece [38] and byte-pair [31] representations. To improve computational performance for tasks involving very long sequences, self-attention could be restricted to considering only a neighborhood of size r in the input sequence centered around the respective output position. This would increase the maximum path length to $O(n/r)$. We plan to investigate this approach further in future work.

A single convolutional layer with kernel width $k < n$ does not connect all pairs of input and output positions. Doing so requires a stack of $O(n/k)$ convolutional layers in the case of contiguous kernels, or $O(log_k(n))$ in the case of dilated convolutions [18], increasing the length of the longest paths between any two positions in the network. Convolutional layers are generally more expensive than recurrent layers, by a factor of k. Separable convolutions [6], however, decrease the complexity considerably, to $O(k \cdot n \cdot d + n \cdot d^2)$. Even with $k = n$, however, the complexity of a separable convolution is equal to the combination of a self-attention layer and a point-wise feed-forward layer, the approach we take in our model.

As side benefit, self-attention could yield more interpretable models. We inspect attention distributions from our models and present and discuss examples in the appendix. Not only do individual attention heads clearly learn to perform different tasks, many appear to exhibit behavior related to the syntactic and semantic structure of the sentences.

5 Training

This section describes the training regime for our models.

5.1 Training Data and Batching

We trained on the standard WMT 2014 English-German dataset consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding [3], which has a shared source-target vocabulary of about 37000 tokens. For English-French, we used the significantly larger WMT 2014 English-French dataset consisting of 36M sentences and split tokens into a 32000 word-piece vocabulary [38]. Sentence pairs were batched together by approximate sequence length. Each training batch contained a set of sentence pairs containing approximately 25000 source tokens and 25000 target tokens.

5.2 Hardware and Schedule

We trained our models on one machine with 8 NVIDIA P100 GPUs. For our base models using the hyperparameters described throughout the paper, each training step took about 0.4 seconds. We trained the base models for a total of 100,000 steps or 12 hours. For our big models,(described on the bottom line of table 3), step time was 1.0 seconds. The big models were trained for 300,000 steps (3.5 days).

5.3 Optimizer

We used the Adam optimizer [20] with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula:

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3}$$

This corresponds to increasing the learning rate linearly for the first $warmup\_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup\_steps = 4000$.

5.4 Regularization

We employ three types of regularization during training:

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Residual Dropout We apply dropout [33] to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply dropout to the sums of the embeddings and the positional encodings in both the encoder and decoder stacks. For the base model, we use a rate of $P_{drop} = 0.1$.

Label Smoothing During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ [36]. This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

6 Results

6.1 Machine Translation

On the WMT 2014 English-to-German translation task, the big transformer model (Transformer (big) in Table 2) outperforms the best previously reported models (including ensembles) by more than 2.0 BLEU, establishing a new state-of-the-art BLEU score of 28.4. The configuration of this model is listed in the bottom line of Table 3. Training took 3.5 days on 8 P100 GPUs. Even our base model surpasses all previously published models and ensembles, at a fraction of the training cost of any of the competitive models.

On the WMT 2014 English-to-French translation task, our big model achieves a BLEU score of 41.0, outperforming all of the previously published single models, at less than $1/4$ the training cost of the previous state-of-the-art model. The Transformer (big) model trained for English-to-French used dropout rate $P_{drop} = 0.1$, instead of 0.3.

For the base models, we used a single model obtained by averaging the last 5 checkpoints, which were written at 10-minute intervals. For the big models, we averaged the last 20 checkpoints. We used beam search with a beam size of 4 and length penalty $\alpha = 0.6$ [38]. These hyperparameters were chosen after experimentation on the development set. We set the maximum output length during inference to input length + 50, but terminate early when possible [38].

Table 2 summarizes our results and compares our translation quality and training costs to other model architectures from the literature. We estimate the number of floating point operations used to train a model by multiplying the training time, the number of GPUs used, and an estimate of the sustained single-precision floating-point capacity of each GPU [5].

6.2 Model Variations

To evaluate the importance of different components of the Transformer, we varied our base model in different ways, measuring the change in performance on English-to-German translation on the

[5]We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

Table 3: Variations on the Transformer architecture. Unlisted values are identical to those of the base model. All metrics are on the English-to-German translation development set, newstest2013. Listed perplexities are per-wordpiece, according to our byte-pair encoding, and should not be compared to per-word perplexities.

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

development set, newstest2013. We used beam search as described in the previous section, but no checkpoint averaging. We present these results in Table 3.

In Table 3 rows (A), we vary the number of attention heads and the attention key and value dimensions, keeping the amount of computation constant, as described in Section 3.2.2. While single-head attention is 0.9 BLEU worse than the best setting, quality also drops off with too many heads.

In Table 3 rows (B), we observe that reducing the attention key size d_k hurts model quality. This suggests that determining compatibility is not easy and that a more sophisticated compatibility function than dot product may be beneficial. We further observe in rows (C) and (D) that, as expected, bigger models are better, and dropout is very helpful in avoiding over-fitting. In row (E) we replace our sinusoidal positional encoding with learned positional embeddings [9], and observe nearly identical results to the base model.

6.3 English Constituency Parsing

To evaluate if the Transformer can generalize to other tasks we performed experiments on English constituency parsing. This task presents specific challenges: the output is subject to strong structural constraints and is significantly longer than the input. Furthermore, RNN sequence-to-sequence models have not been able to attain state-of-the-art results in small-data regimes [37].

We trained a 4-layer transformer with $d_{model} = 1024$ on the Wall Street Journal (WSJ) portion of the Penn Treebank [25], about 40K training sentences. We also trained it in a semi-supervised setting, using the larger high-confidence and BerkleyParser corpora from with approximately 17M sentences [37]. We used a vocabulary of 16K tokens for the WSJ only setting and a vocabulary of 32K tokens for the semi-supervised setting.

We performed only a small number of experiments to select the dropout, both attention and residual (section 5.4), learning rates and beam size on the Section 22 development set, all other parameters remained unchanged from the English-to-German base translation model. During inference, we

Table 4: The Transformer generalizes well to English constituency parsing (Results are on Section 23 of WSJ)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJ only, discriminative	88.3
Petrov et al. (2006) [29]	WSJ only, discriminative	90.4
Zhu et al. (2013) [40]	WSJ only, discriminative	90.4
Dyer et al. (2016) [8]	WSJ only, discriminative	91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

increased the maximum output length to input length + 300. We used a beam size of 21 and $\alpha = 0.3$ for both WSJ only and the semi-supervised setting.

Our results in Table 4 show that despite the lack of task-specific tuning our model performs surprisingly well, yielding better results than all previously reported models with the exception of the Recurrent Neural Network Grammar [8].

In contrast to RNN sequence-to-sequence models [37], the Transformer outperforms the Berkeley-Parser [29] even when training only on the WSJ training set of 40K sentences.

7 Conclusion

In this work, we presented the Transformer, the first sequence transduction model based entirely on attention, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

For translation tasks, the Transformer can be trained significantly faster than architectures based on recurrent or convolutional layers. On both WMT 2014 English-to-German and WMT 2014 English-to-French translation tasks, we achieve a new state of the art. In the former task our best model outperforms even all previously reported ensembles.

We are excited about the future of attention-based models and plan to apply them to other tasks. We plan to extend the Transformer to problems involving input and output modalities other than text and to investigate local, restricted attention mechanisms to efficiently handle large inputs and outputs such as images, audio and video. Making generation less sequential is another research goals of ours.

The code we used to train and evaluate our models is available at `https://github.com/tensorflow/tensor2tensor`.

Acknowledgements We are grateful to Nal Kalchbrenner and Stephan Gouws for their fruitful comments, corrections and inspiration.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017.

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Zhongqiang Huang and Mary Harper. Self-training PCFG grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 832–841. ACL, August 2009.

[15] Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

[16] Łukasz Kaiser and Samy Bengio. Can active memory replace attention? In *Advances in Neural Information Processing Systems, (NIPS)*, 2016.

[17] Łukasz Kaiser and Ilya Sutskever. Neural GPUs learn algorithms. In *International Conference on Learning Representations (ICLR)*, 2016.

[18] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099v2*, 2017.

[19] Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *International Conference on Learning Representations*, 2017.

[20] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[21] Oleksii Kuchaiev and Boris Ginsburg. Factorization tricks for LSTM networks. *arXiv preprint arXiv:1703.10722*, 2017.

[22] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[25] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[26] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159. ACL, June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 433–440. ACL, July 2006.

[30] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

Attention Visualizations

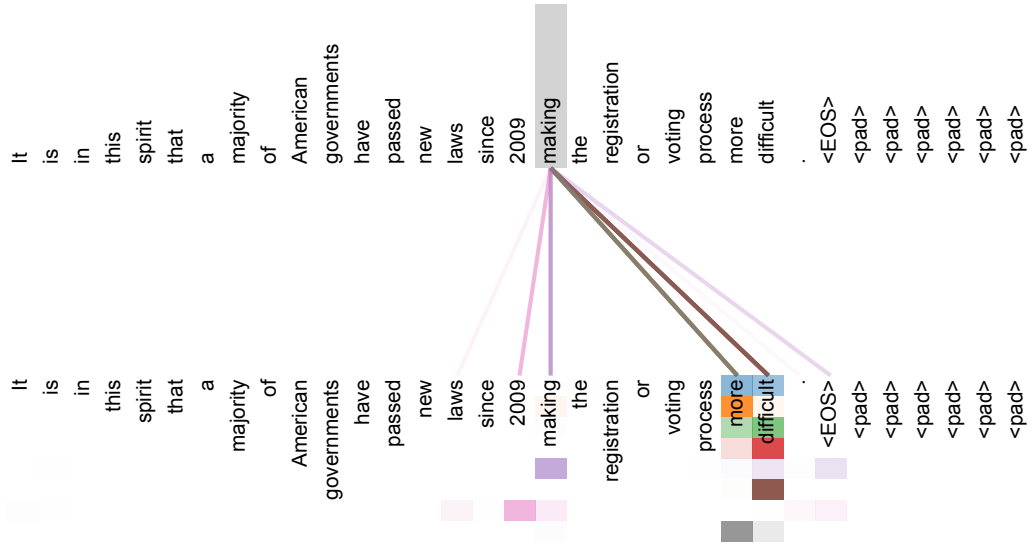


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult'. Attentions here shown only for the word 'making'. Different colors represent different heads. Best viewed in color.

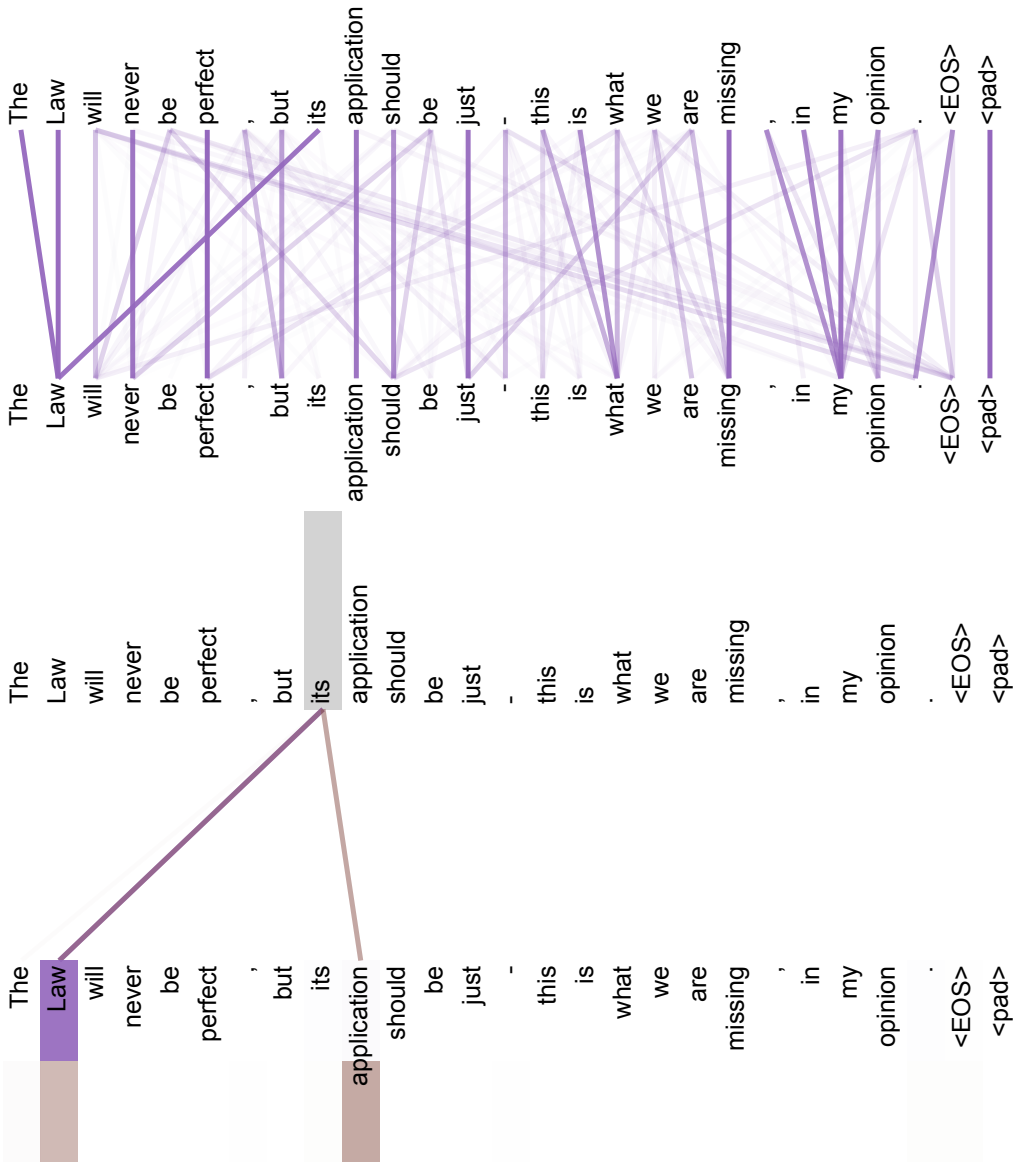


Figure 4: Two attention heads, also in layer 5 of 6, apparently involved in anaphora resolution. Top: Full attentions for head 5. Bottom: Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word.

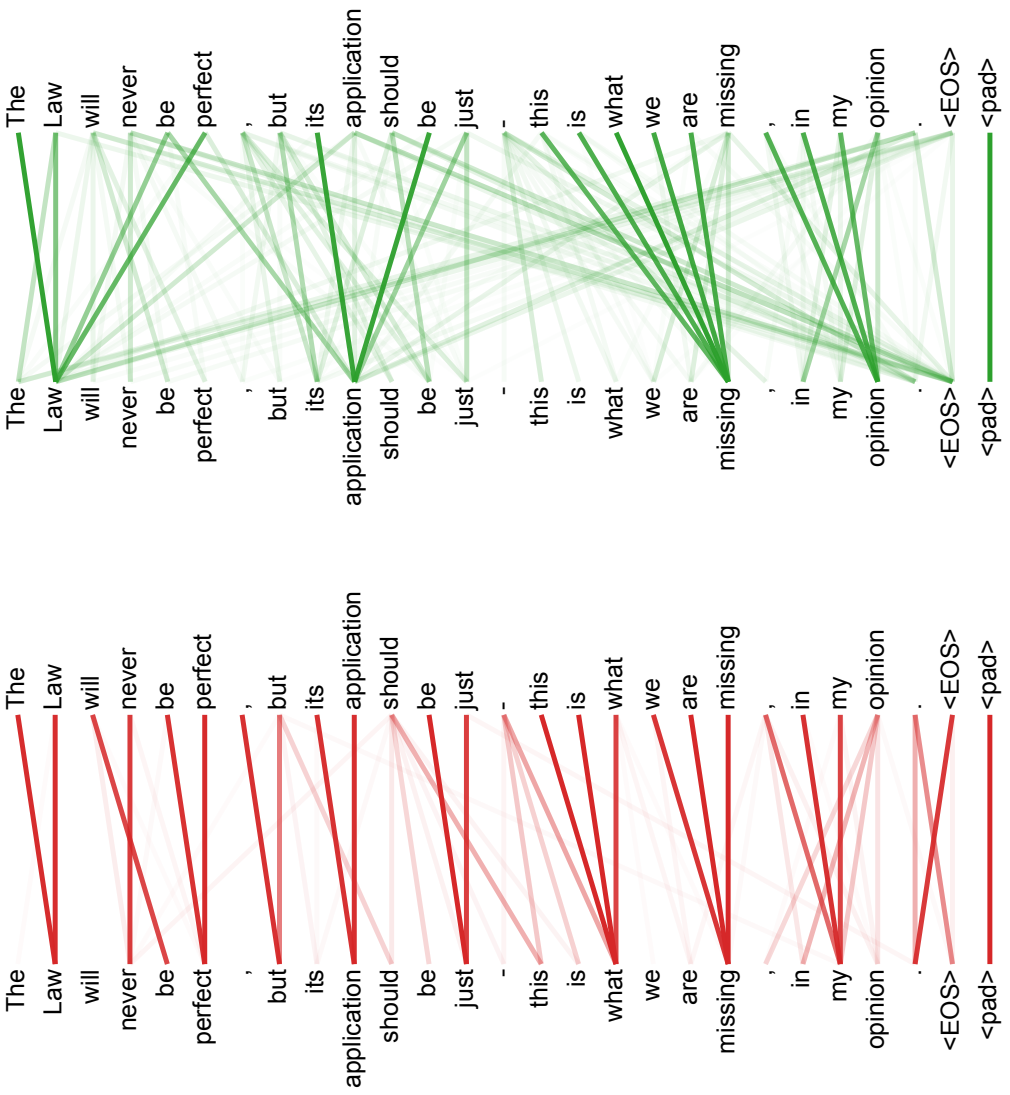


Figure 5: Many of the attention heads exhibit behaviour that seems related to the structure of the sentence. We give two such examples above, from two different heads from the encoder self-attention at layer 5 of 6. The heads clearly learned to perform different tasks.