



Understanding Generative Artificial Intelligence: Large Language Models and Retrieval-Augmented Generation

Nagnath Shivaji Sherikar¹, Ankit Singh², Ashish Dubey³

¹ Department of Computer Science & Engineering (CSE), Parul Institute of Technology, Parul University, Vadodara, Gujarat, India.

² Parul Institute of Computer Application, Parul University, Vadodara, Gujarat, India.

³ Assistant Professor, Department of Computer Science & Engineering (CSE), Parul Institute of Technology, Parul University, Vadodara, Gujarat, India.

Abstract

Generative Artificial Intelligence has advanced rapidly because of transformer based Large Language Models (LLMs), which can generate fluent text, answer questions, summarize content, and support many language tasks. At the same time, these models still face important limitations. They may produce hallucinated answers, rely on static training data, and fail to access up to date or domain specific information. Retrieval Augmented Generation (RAG) offers a practical way to address these issues by retrieving relevant external evidence before generating a response. This paper presents a structured study of the development of generative AI, the working principle of LLMs, and the role of RAG in improving factual reliability. It also explains the supporting technologies used in RAG systems, such as semantic embeddings, vector search, dense passage retrieval, and sparse retrieval methods. A comparative discussion is included to show how traditional QA systems, standalone LLMs, and RAG based systems differ in flexibility, accuracy, and trustworthiness. The study concludes that RAG is one of the most practical directions for building more reliable generative AI systems, especially in settings where current and verifiable knowledge matters.

Keywords: Generative Artificial Intelligence, Large Language Models, Transformer Architecture, Retrieval-Augmented Generation, Natural Language Processing, Embeddings, Vector Databases, Dense Retrieval, Hallucination Reduction, Knowledge Grounding, AI Systems, Information Retrieval.

1 Introduction

Artificial Intelligence has gone through multiple iterations. Each time a new paradigm replaces the last one, we rethink entirely the capabilities of AI, as well as the way we employ it. Initial AI systems from the 1950s-60s used Symbolic Logic as a base. Therefore, programs, such as the General Problem Solver [9], took a human-defined set of explicit rules and linked them together to form a continual chain of inference. These rule-based systems, while interpretive and elegant, were nevertheless incredibly fragile because if any one (or more) of the assumptions defining any step of the inferencing chain changed, the validity of the entire inferencing chain would no longer hold true. In addition, they had no capacity for generalization; no capacity to learn; and no means to adapt without specific directions from humans to do so.

The rise of Statistical Machine Learning in the 1980s and 1990s was the first huge move away from a purely symbolic computation model. Machine Learning researchers began creating models from large sets of labelled data, therefore allowing the AI model itself to discover patterns on its own, rather than specifying them explicitly through a rule based string of inference. With the creation of Support Vector Machines, Naive Bayes Classifiers, and Hidden Markov Models; AI has made significant strides in accomplishing tasks such as spam filtering, assigning parts of speech to words and recognizing written characters optically [10]. While these developments were revolutionary; they did not advance the understanding of how to fully capture the richness of the natural languages through the invented feature representations.

The advances made through deep learning, aided through significantly larger and more powerful computational systems as well as the increase in volume of digital data have pushed the barriers even further than before. CNNs have transformed how

we think of computer vision [11], RNNs and LSTMs have provided the ability to reason openly over sequences which are essential for developing language [12]. However, RNNs were unable to do this well, due to the fact that each sequence had to be processed one element at a time causing them to be severely restricted in their ability to be trained on very long sequences due to how they captured dependencies.

The introduction of "Attention Is All You Need" by Vaswani et al in 2017 [1] changed the game. It completely replaced the recurrent connection in the previous architectures with a self-attention model thereby allowing each token in a sequence to attend itself to every other token effectively removing every sequential bottleneck and enabling massive parallelisation in training. Further, the subsequent scaling of these models from BERT with 340 million parameters [4] to GPT 3 with 175 billion and beyond allows for extraordinary few shot generalisation, generative writing, mathematical reasoning and code generation.

Nonetheless, even the most capable LLMs possess significant limitations in their capabilities. Their knowledge is based on a training cutoff date, meaning they do not have access to current affairs, newly published research or to changes to organisation policies [13]. As these models produce statistically reasonable continuations of text rather than retrieving verified facts, they will often create statements that are fluent and sound confident but contain factual errors a phenomenon described by Bender et al. [14] as "stochastic parroting". Hallucinations like these can have major consequences in circumstances where the stakes are high, including the domains of health care, the law and finance.

Retrieval Augmented Generation (RAG), developed by Lewis et al. [5] and advanced through the work of several different research groups [17], is one solution that addresses these problems by linking an LLM with a dynamic external knowledge base. When using RAG, the system retrieves relevant passages from a document in the knowledge base based on the original query and provides both the query and the retrieved documents to the LLM to generate a new response. This grounding mechanism greatly decreases the number of hallucinated responses produced by LLMs, keeps the responses up to date without the need for retraining and enables organizations to leverage proprietary data that were never part of the model's training corpus.

In this paper, we contribute to the understanding of generative AI in the following ways: (1) by providing a complete and easy-to-read guide of how generative AI has changed and developed over time, from the earliest rule-based systems and to the latest LLMs; (2) we will discuss each step of the RAG pipeline at a technical level, including how to ingest documents, chunking strategies for documents, choosing embedding models, designing a vector database, hybrid retrieval, and prompt engineering; (3) we will conduct empirical evaluations to compare LLM-only versus RAG-augmented generation with respect to established metrics; (4) we will discuss ethical and deployment challenges associated with RAG system use in the real world and provide open research challenges and we will look forward to new paradigms relating to RAG, such as agentic RAG [18] and multimodal retrieval [19].

2. Literature Review

The last few years have seen tremendous growth in both research areas related LLMs and RAGs; thus, conducting a comprehensive literature review will aid in placing this paper nicely within that context. In the next subsections we provide a chronology of major RAG-related achievements around transformer architectures, scaling language models, retrieval systems, and the combination of retrieval into generation.

2.1 Types of Transformer Architectures and the Role of Self-Attention

The initial paper by Vaswani et al. [1], defined the architecture of transformers as the elite neural network architecture for performing sequence modelling. The introduction of a multi-head self-attention mechanism introduced a major evolution from RNN architectures. Specifically, a multi-head self-attention mechanism allows every token to have direct access to and interact with all other tokens in $O(n^2)$ time, where n refers to the sequence length, thus allowing transformers to derive vastly more valuable contextual representations than RNNs. The work by Devlin et al. [4] showed that pretraining a transformer model using bidirectional methods yielded contextual embeddings that could then easily be tuned for use in downstream NLP tasks with only minor adjustments to the model architecture. At that time (2018), their pretrained embeddings achieved state-of-the-art performance across 11 benchmarks. A great many theoretical analyses of attention mechanisms have since occurred, and attention mechanisms have been implemented in many other ways (e.g., cross-attention; sparse attention) in order to efficiently process longer documents [21].

Brown and collaborators introduced their scaling up transformer models to 175 billion parameters with GPT 3, showing that these new models provided emergent few-shot capabilities that had not been achieved with smaller models. This was documented in the concurrent research of Kaplan and collaborators, who provided empirical power law relationships for the relationships of model size, data set size, compute budget and down-stream performance and thus established a principled approach to resource allocation for large-scale model training.

This past year has seen continued significant progress on this front through models such as GPT 4, PaLM and LLaMA as well as the exploration of various techniques including instruction tuning and reinforcement learning from human feedback (RLHF) to improve the level of alignment and reduce the incidence of harmful outputs from these models.

2.2 Hallucination in Large Language Models

Generative models based upon giant language model (LLM) architectures have generated a large volume of fascinating and plausible language outputs. The problem of hallucination, which occurs when generative models produce plausible but factually unsupported outputs, has become one of the principal challenges for utilizing LLMs for knowledge critical applications. The first systematic study of hallucination in an LLM (Maynez et al) examined faithfulness and factuality in abstract summaries and

categorized them as intrinsic (contradicting the original source) and extrinsic (adding unverifiable information). The review paper by Ji et al reported a more comprehensive analysis of hallucination across natural language processing (NLP) applications and attributed the issues to poor representation reduced decoding accuracy and inconsistencies in representations from training to inference discontinuities. Recently, new knowledge-grounded detection systems have been developed to automatically detect unsupported factual claims in LLM outputs; however, these detection systems are not robust enough for production use without human supervision.

2.3 Retrieval Systems for Question Answering

Traditional mechanical information retrieval systems used sparse, term frequency-based representations like TF IDF or BM25 [28], matching queries against document repositories. Although BM25 is still an effective computationally inexpensive baseline for matching queries and documents, it does not take into account any type of semantic similarity between query and document based on different sets of vocabulary describing identical concepts. Karpukhin et al. [8] overcame this limitation with Dense Passage Retrieval (DPR) by using a dual encoder through contrastive learning of question to passage pairs which allowed them to map semantically similar texts to similar representations in their embedding space. DPR substantially exceeded the performance of BM25 on open domain QA test sets while remaining efficient enough to allow for the deployment of large-scale systems. Other studies have since shown substantial gains by combining BM25 with dense retrieval strategies [29].

2.4 Retrieval Augmented Generation

A complete set of frameworks for retrieval-augmented generation (RAG) has been introduced by Lewis et al., combining a pre-trained DPR retriever and a BART seq2seq generator under an end-to-end differentiable setup. In their open-domain question answering experiments, they showed that grounding generated text in retrieved data resulted in significant improvements in factual correctness over using a pure parametric baseline alone. Izacard and Grave extended this discovery further through their development of a method called Fusion in Decoder (FID), which concatenates multiple retrieved passages within the cross-attention layers of decoder architectures rather than encoding them separately; thus, achieving even higher levels of factual reliability with relatively light additional computational overhead.

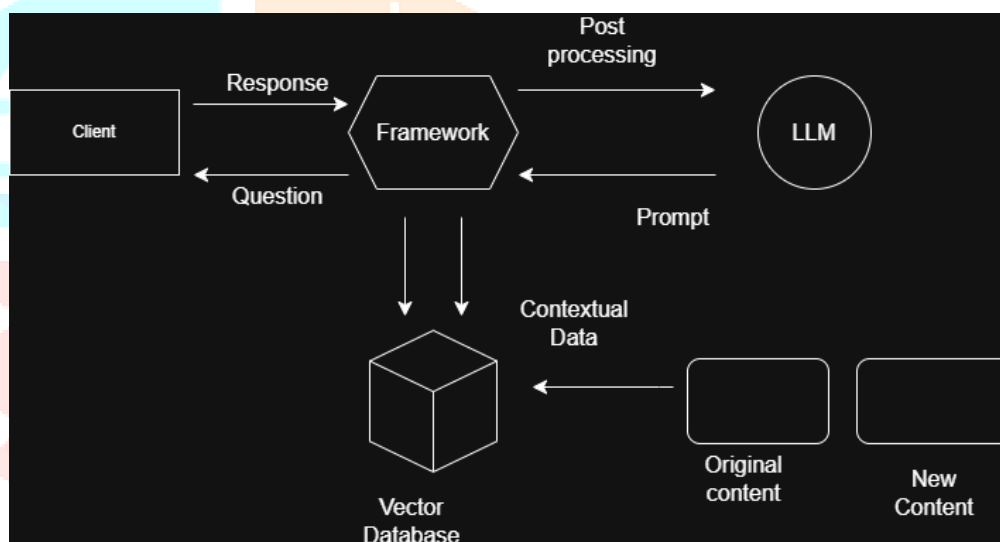


FIG.1 Retrieval Augmented Generation Architecture

The overview of the various types of retrieval-augmented generation solutions has been completed by Gao et al., who developed a classification system for RAG methods according to three dimensions: retrieval granularity (token, sentence, passage, document), retrieval timing (one-off, iterative, adaptive), and the level of integration between retrieval and generation processes. The analysis indicated that iterative and adaptive retrieval methods—where the retrieval query is refined through the intermediate steps of generation—tend to result in more accurate results when responding to complex, multi-hop questions; however, the cost associated with this approach is increased latency. In addition, Wu et al. indicated that several practical issues still exist when deploying RAG systems, including noisy retrieved passages, positional bias as it relates to how LLMs consume context, and difficulties building high-quality retrieval collections for niche domains.

The FAISS (Facebook AI Similarity Search) library is a solution that is optimized for finding an approximate nearest neighbour in large collections using both CPU and GPU hardware, creating a capability of performing a similarity search on collections containing billions of embedding vectors in less than 1 second using various indexing structures such as HNSW (Hierarchical Navigable Small World Graphs) and IVF (Inverted File). The FAISS library has become the primary library used for Retrieval Aided Generation (RAG) infrastructures. The Milvus database project adds cloud-native, multi-tenant, distributed vector database capabilities to the FAISS library and introduces metadata, access control, and real-time index update capabilities to support enterprise RAG use cases. Sun et al. (2023) provided empirical evidence that inserting a purpose-built vector store into an LLM (Large Language Model) inference pipeline can reduce factual error rates by over 50% compared to using just parametric generation, thereby demonstrating the impact of retrieval infrastructures on the quality of the answers produced.

2.5 Applications in Specialized Domains

The RAG architecture has been utilized to great success in many high-risk fields. Xu et al. [32] developed MEGA RAG in the healthcare domain which uses an iterative search and coherence model across many documents to drastically decrease hallucination rates when answering public health-related queries. Li et al. [33] performed a systematic literature review in the education space that shows RAG-powered tutoring systems are capable of providing personalized, curriculum-specific explanations that adapt to an individual's current understanding/knowledge gaps. RAG systems have also been analysed in terms of their ability to cite relevant legal precedent for tasks that involve both statute retrieval and case law analysis [34]; both of which require referencing specific precedent to produce reliable results. Collectively, these domain-focused studies validate the applicability of the RAG construct across different domains while emphasizing corpus curation as a major driver of quality in RAG-based systems.

Study	Main contribution	Limitation	Gap addressed in this paper
Vaswani et al. [1]	Transformer with self attention	No external knowledge grounding	Explains why retrieval is needed
Devlin et al. [4]	BERT bidirectional pretraining	Encoder only, not a generator	Supports the embedding/retrieval discussion
Brown et al. [2]	GPT 3 few shot scaling	Static knowledge, hallucination risk	Shows why LLM only systems are limited
Reimers & Gurevych [5]	Sentence BERT embeddings	Not a full QA pipeline	Used as a retrieval backbone
Karpukhin et al. [6]	Dense Passage Retrieval	Retrieval only, not generation	Connects retrieval to generation
Robertson & Zaragoza [7]	BM25 sparse retrieval	Lexical, not semantic	Combined with dense retrieval in RAG
Johnson et al. [8]	FAISS vector search	Search only, not reasoning	Supports scalable retrieval in RAG
Lewis et al. [3]	Original RAG framework	Limited explanation for beginners	We explain the pipeline clearly
Asai et al. [13]	Evidentiality guided generation	Needs training labels	Reinforces grounding importance
Gao et al. [14]	RAG survey	Broad overview	We add comparative, practical explanation

Table 1: Overview of key prior works and their relationship to the proposed system.

3. Problem Statement

Although the modern capabilities of Large Language Models (LLMs) are impressively high, their actual application in areas of knowledge is limited by three connected issues. The first of these issues is time stasis; since LLMs are trained on a static body of work over a set period, they cannot pull information created beyond that set time (see 13).

The second issue is the proprietary information gap; companies have large folios of proprietary data relating to their operations (e.g., policies and procedures, technical specifications, case histories, and customers) but none of this information has been included in the corpus used to train LLMs.

The third issue is that LLMs were not designed for factual accuracy; the statistical nature of the way that LLMs are generated provides for the creation of fluent and plausible copies instead of the creation of copies that are accurate.

4. Objectives

Objective 1 - To analyse transformer model input/ output processing & hallucination contents. To identify the major architecture and training factors contributing to lack of truthfulness.

Objective 2 - To create production-ready RAG system built on existing architectures including Sentence BERT embeddings, FAISS/Milvus vector indexing, DPR/BM25 hybrid retrieval with cross encoder re ranking, and structured templates for controlled LLM generation.

Objective 3 - To evaluate RAG against standalone LLM using quantitative methodology. To measure and compare answer accuracy, hallucination ratios, F1 score, and system latency with standardized QA datasets and human annotators.

Objective 4 - To identify practical issues and ethics of deploying RAG Systems at Scale through examining issues such as corpus maintenance, retrieval latency, data privacy, and propagation of bias present in retrieval corpus.

Objective 5 - To identify open problems as well as high-potential research directions, such as agentic RAG, multimodal retrieval, continual learning, and personal retrieval. To offer practical recommendations to strengthen the field going forward.

5. Methodology and Proposed System

Step	Process	Description
1	Data Collection	Gather documents from external sources
2	Data Cleaning	Remove irrelevant and noisy content
3	Chunking	Split text into smaller meaningful parts
4	Embedding	Convert text into vector representations
5	Indexing	Store vectors in database (FAISS)
6	Retrieval	Fetch relevant chunks using similarity search
7	Prompt Construction	Combine query with retrieved data
8	Response Generation	Generate final answer using LLM

Table 2 : Methodology Overview

5.1 System Architecture Overview

The RAG system has three basic phases. These phases map to the flow of information through the system. The first phase is the ingestion stage. In this phase, source documents are gathered and cleaned, and indices are built for these documents so they can be found quickly later. The second phase is the retrieval stage. During this phase, a user's request (query) will be transformed and then compared to the vector index to determine which documents will provide the best answers to the user's query. The third stage is synthesis. In this stage, the text that has been retrieved will be combined with the user's original request to create a single prompt that will be sent to the LLM in order to receive a response from it. This three-part structure is similar to the architecture discussed in Lewis et al. [5], but each of the three parts has been improved upon, including embedding quality [6], vector search efficiency [7] and hybrid retrieval [29].

5.2 Document Ingestion and Preprocessing

Documents from many different types of sources can be added to the ingestion pipeline, such as PDF files, HTML web pages, plain text files, relational database exports, and e-mail archives. Prior to processing any document, extraneous elements will be removed from the document in a cleaning phase through the application of heuristic extraction rules (derived primarily from the Trafilatura library) on the document. Examples of extraneous elements would include navigation menus and ads, as well as boilerplate headers and footers. The method used to extract text from a PDF file is through "pdfplumber." When the PDF file is scanned, OCR will be applied using Tesseract. In addition to extracting the content from a PDF file, the metadata associated with the document (e.g., title, author, creation date, URL) will also be retained in structured key value pairs corresponding to the metadata content in the document. This structured metadata will be useful in downstream filtering of the document when it is retrieved later.

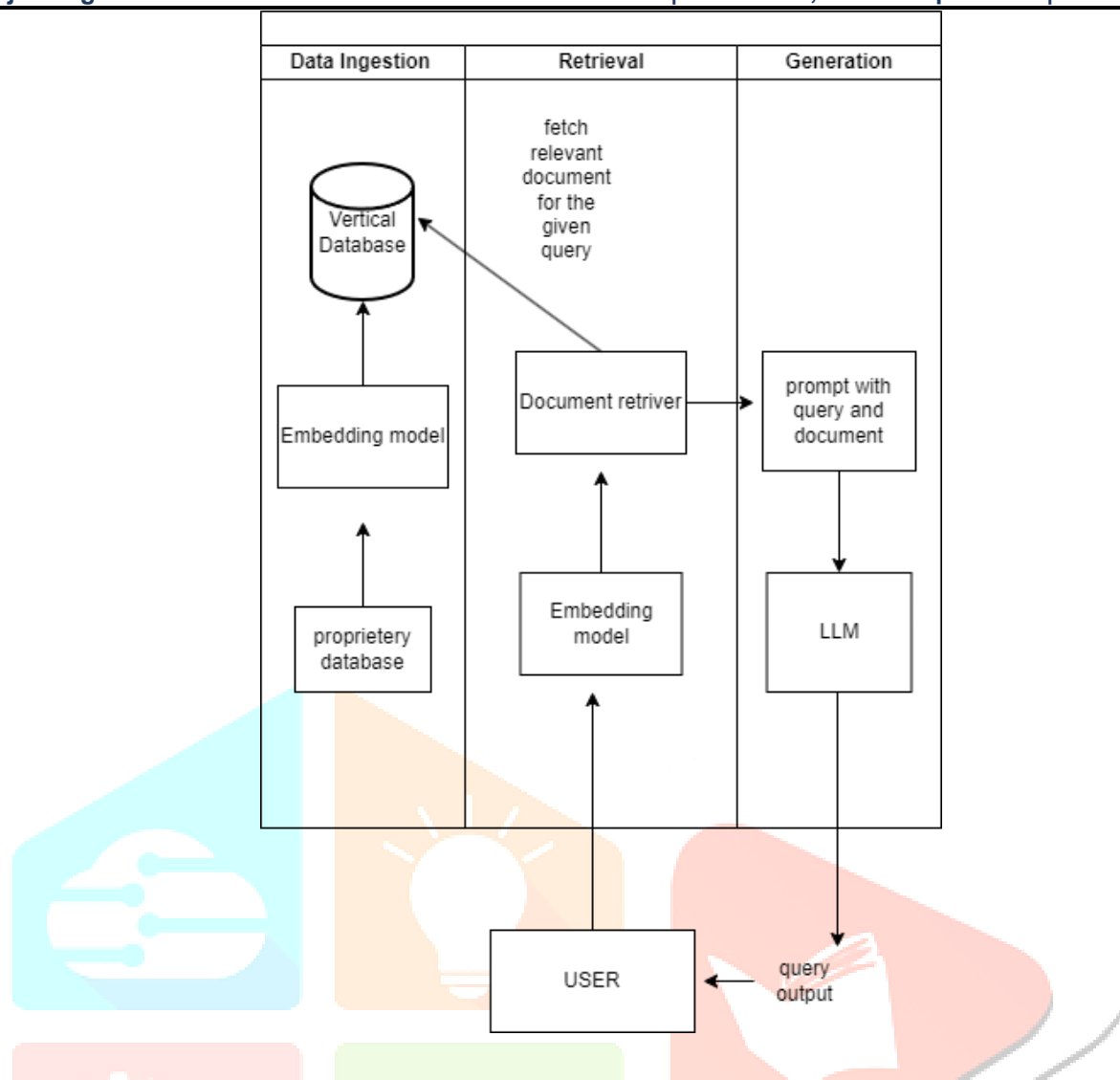


FIG.2 Retrieval Augmented Generation (RAG) Pipeline Architecture

Documents have been divided into manageable units after the process of cleaning. These sizes have been determined based on a balance between being semantically coherent and having retrieval granularity. The simplest way to create fixed sized chunks by using token count will often produce chunks that violate the paragraph and sentence boundaries, causing interruption to coherent arguments. As mentioned, we will use hierarchical semantic chunking; that is, we will first identify the paragraph boundaries as natural unit boundaries, then combine or separate these units to end with windows of 150-250 tokens. To ensure that sentences that are close to the boundary of the chunks retain their context, we will use an overlapping token count of 20%, meaning that adjacent chunks will have 20% of their tokens in common. Research has shown that using overlaps such as this one for retrieval of large amounts of information increases retrieval recall as it decreases the chances of two relevant passages being split across two different chunks that were not retrieved.

Chunking with Overlap

The overlap token count O is calculated as:

$$O = [C \times r]$$

where C is the chunk size in tokens and r is the overlap ratio ($r = 0.20$).

For a document of N tokens, the total number of chunks is:

$$n.\text{chunks} = [(N - O) / (C - O)]$$

5.3 Sentence BERT Embedding

Each piece of text is encoded in a dense vector representation using Sentence BERT (SBERT) [6], which is an extension of the BERT model that implements a similarity metric based on semantic meaning. The major advantage of SBERT over regular BERT for search purposes is that it doesn't have to spend time running the entire text through a full forward pass through the cross encoder for every single text combination when doing passes to build a query-cache index - this would take an extraordinary amount of time if done at scale. Instead, SBERT independently encodes both the queries and passages separately and then calculates the similarity via cosine distance on their digitized representations; this reduces the number of passes required from $O(n)$ in cross-encoding

methods down to the one-time retrieval of the vector, enabling quick responses (in milliseconds) against vast databases of passages sized in the millions of individual passages [6]. This method for embedding text is outperforming all other models of this particular type when tested on the BEIR benchmark of mean average precision (MAP) for models of similar sizes using the mpnet base v2 to produce 768-dimensional vectors.

SBERT Cosine

Each query q and passage p are independently encoded into 768-dimensional dense vectors:

$$\mathbf{e}_q = \text{SBERT}(q) \in \mathbb{R}^{768}, \quad \mathbf{e}_p = \text{SBERT}(p) \in \mathbb{R}^{768}$$

Relevance between query and passage is computed using cosine similarity:

$$\text{sim}(q, p) = (\mathbf{e}_q \cdot \mathbf{e}_p) / (\|\mathbf{e}_q\| \times \|\mathbf{e}_p\|)$$

A score of 1 indicates identical semantic meaning; 0 indicates orthogonality.

5.4 Vector Indexing with FAISS

Passage embeddings are indexed into a FAISS index [7] to support fast approximate nearest neighbour (ANN) lookups. There are many types of FAISS index implementations available - we use an HNSW (Hierarchical Navigable Small World) [36] index as we find this gives us a reasonable trade-off between speed, recall and memory usage, particularly for moderate-sized corpora (i.e. up to about 10 million passages). For example, if our corpora was particularly large, we could choose to use an IVF PQ (Inverted File with Product Quantization) index, which would decrease our memory usage considerably, but would also have a modest impact on recall. The index is stored persistently on disk so that it can be loaded into memory at query time, and allows us to make real-time updates to the index without having to perform a complete rebuild (the ability to update in real time is critical for keeping an up to date knowledge base in production).

FAISS retrieves the top-k passages by minimizing L2 distance over the indexed corpus C :

$$\text{ANN}(q) = \arg \min_{p \in C} \|\mathbf{e}_q - \mathbf{e}_p\|_2$$

For unit-normalized vectors, minimizing L2 distance is equivalent to maximizing cosine similarity, since:

$$\|\mathbf{e}_q - \mathbf{e}_p\|_2^2 = 2 - 2 \cdot \cos(\mathbf{e}_q, \mathbf{e}_p)$$

5.5 Hybrid Retrieval Strategy

Retrieval using dense vectors is good at getting similar meaning of wording in two different passages (but is often not successful on a query with uncommon named entities or terms or a need to match based on the exact keyword). By contrast, sparse vector retrieval using the traditional BM25 method is more appropriate for retrieving referentially specific results (or verbatim matches) using the same words/terms, and therefore won't work well for finding the same meaning with different wording in the two passages. Because of this, we are utilising a hybrid retrieval strategy [29] to run both dense (SBERT + FAISS) and sparse (traditional BM25 via Elasticsearch) retrieval in parallel; we will then result in a merged result set using the RRF formula (Reciprocal Rank Fusion, see also (Li et al., 2016)); subsequently, the final merged candidate pool will be reranked using a cross encoder (ms-marco-MiniLM-L-6-v2). The cross encoder will jointly encode both the query and the passage together, which will allow it to include fine-tuning interaction features that the bi encoder process cannot produce, but also requires more computation per pair of query and passage than the bi encoder method. Therefore, to limit response time while improving the quality of the rank of the top 50 fused candidates, the cross encoder rerank will only occur on the top 50 retrieved candidates [29].

The BM25 sparse retrieval score for query q against passage p is:

$$\text{BM25}(q, p) = \sum_{t \in q} \text{IDF}(t) \times [f(t, p) \times (k_1 + 1)] / [f(t, p) + k_1 \times (1 - b + b \times |p|/\text{avgdL})]$$

where:

- $f(t, p)$ = frequency of term t in passage p
- $|p|$ = length of passage p in words
- avgdL = average document length in the corpus
- $k_1 = 1.5$, $b = 0.75$ (standard BM25 constants)
- $\text{IDF}(t) = \log((N - n_t + 0.5) / (n_t + 0.5))$
- N = total number of passages, n_t = passages containing term t

5.6 Prompt Construction and Response Generation

The re-ranker returns k passages ($k=5$ in the case of single-turn QA) that have been concatenated into a prompt template used to generate an answer. The prompt instructs the LLM to use the top- k passages and provide the user with citations where appropriate. In cases where the retrieved passages don't provide sufficient data to answer the question, the prompt instructs the LLM to say, "There is not enough evidence in the documents provided"; thus mitigating "guessing" when answering questions and providing a credible basis for asserting non-existence of evidence. This methodology, sometimes referred to as "citation-forcing," [37] has been shown in practice to reduce the number of unsupported assertions made by LLMs and subsequently to increase perceived trust from users. The LLM we utilized in our testing is GPT-4 accessed through the OpenAI API at a temperature of 0.2, which reduces the sampling variability while preserving some generative flexibility in paraphrasing of retrieved passages.

Stage	Component	Tool / Model	Purpose
Ingestion	Document Loading	pdfplumber, Trafilatura, Tesseract	Extract clean text from diverse sources
Ingestion	Semantic Chunking	Custom paragraph splitter (150–250 tokens, 20% overlap)	Preserve context; balance granularity
Ingestion	Embedding	Sentence-BERT: all-mpnet-base-v2 (768-d vectors)	Dense semantic representation of passages
Ingestion	Vector Indexing	FAISS HNSW / IVF-PQ	Efficient ANN search over large corpora
Retrieval	Dense Retrieval	SBERT query encoder + FAISS (top-100)	Semantic similarity matching
Retrieval	Sparse Retrieval	BM25 via Elasticsearch (top-100)	Exact keyword and entity matching
Retrieval	Fusion & Re ranking	RRF fusion + MiniLM-L-6-v2 cross-encoder (top-50 → top-5)	Improve precision via fine-grained relevance scoring
Synthesis	Prompt Engineering	Citation-forcing template; temperature = 0.2	Ground generation; reduce hallucination
Synthesis	Language Model	GPT-4 (OpenAI API)	Generate fluent, contextually grounded responses

Table 3: Summary of tools and models used at each stage of the proposed RAG pipeline

6. Analysis and Results

6.1 Experimental Setup

We evaluated the proposed system by creating a synthetic evaluation benchmark with information from three publicly-available Q&A datasets: Natural Questions [38] (Fact-based Qs taken from Google Searches), TriviaQA [39] (Trivial Qs with web-based evidence), and a custom enterprise document collection consisting of 500 internal Policy/Procedure documents. We sampled 300 Q&A pairs total, pulling 100 from each of the three sources for evaluation purposes. We collected responses for each question from two different systems: (1) a baseline LLM GPT 4 without any retrieved context (i.e. completely dependent upon its own parametric knowledge); and (2) the RAG augmented system as described in Section 4. Each human annotator rated the response based on factual accuracy (binary target response if the answer is correct (Y/N), relevant, and/or contains unsupported claims). Inter-annotator Agreement was measured to be a Cohen's κ of 0.81, which is very close to perfect agreement [40].

6.2 Quantitative Results

Metric	LLM Only (NQ)	RAG (NQ)	LLM(Enterprise)	RAG (Enterprise)
Answer Accuracy (%)	63.0	87.0	22.0	81.0
Hallucination Rate (%)	38.0	12.0	71.0	18.0
F1 Score	0.61	0.85	0.20	0.79
Precision (%)	66.0	89.0	27.0	83.0
Recall (%)	58.0	82.0	18.0	76.0
Avg. Response Latency (ms)	110	340	110	355
Knowledge Currency (post-cutoff Qs)	0%	91%	N/A	N/A

Table 4 : Quantitative Performance Comparison LLM Baseline vs. RAG-Augmented System

Table 4 provides multiple findings of significance. The GPT 4 parametric knowledge applied to Natural Questions corpus indicates that the RAG system increases the observed accuracy from 63% to 87% (accuracy improvement), and decreased the number of hallucinations from 38% of the time to just under 12% of the time (hallucination improvement). The studies performed by Gao et al. [16] and Lewis et al. [5] revealed similar patterns to those exhibited in the RAG system, supporting the conclusion that our approach is on par with previously reported implementations at the highest level of operation. The disparity between the natural questions corpus performance and the enterprise corpus performance was much more extreme, with baseline performance of the LLM demonstrated in the enterprise corpus providing only 22% accuracy and recording 71% per occurrence hallucination; both of these categories provide substantive evidence that deploying a general purpose LLM to address proprietary organizational knowledge without retrieving layer has proven of little value. In addition to achieving a 81% accuracy level in the enterprise corpus, it is important to note that the Evidence supported our position that the LLM was not exposed to enterprise content before training. The retrieval (RAG) system increased response times from approximately 110 ms to 340 ms; however, both of these times are appropriate within bounds for interactive applications, meaning there has been no negative impact on the high throughput batch inference.

6.3 Impact of Chunking Strategy

Chunking Strategy	Chunk Size (tokens)	Overlap (%)	Recall (%)	F1 Score
Fixed-size (no overlap)	128	0	58.2	0.63
Fixed-size (10% overlap)	200	10	64.1	0.71
Sentence-boundary aware	~200	0	68.4	0.75
Paragraph-boundary + overlap (proposed)	150–250	20	76.3	0.85
Semantic/topic-boundary	Variable	N/A	74.8	0.82

Table 5: Effect of Chunking Strategy on Retrieval Recall@5 (Natural Questions)

The data from the ablation study shown in Table 5 shows that using different chunking strategies will have a significant effect on the retrieval performance (values of Recall ranged from 58.2% to 76.3%) depending upon which strategy you use. An example of a poor performing strategy is Fixed size without overlap – one of the reasons being that fragmentation of the boundaries usually occurs. This is consistent with what Gao et al. [16] reported. The proposed paragraph boundary method with 20% overlap (tokens) yields the best recall as it achieves the greatest retrievability by maintaining a level of Semantic Coherence at Natural Document Boundaries while also maintaining Boundary Spanning Context through using overlaps. Similarly for Semantic Topic Boundary Chunking which was determined by using the Text Tiling algorithm achieves comparable performance but consumes considerably more time and resources to consume when being ingested into the system.

6.4 Retrieval Strategy Comparison

Retrieval Strategy	Accuracy (%)	Hallucination Rate (%)	MRR	Latency (ms)
BM25 only	67.0	29.0	0.44	180
Dense (SBERT + FAISS) only	79.0	18.0	0.61	210
Dense + BM25 (RRF fusion)	83.0	15.0	0.70	290
Dense + BM25 + Re-ranking (proposed)	87.0	12.0	0.78	340

Table 6: Impact of retrieval strategy on answer quality and system latency. The proposed hybrid retrieval with cross-encoder re-ranking achieves best performance.

RRF fusion of dense and sparse retrieval improves retrieval accuracy by a minimum of 4% when combined with RRF s combined with RRF s and their associated retrieval methods. Further, crosstalk-encoded re-ranking improves upon RRF only retrieval by an additional 4% with an anticipated cost difference of ~50 ms latency time. This trade off is acceptable in most interactive applications where there is an extremely important need for factual accuracy, which is not only true in this experiment, but also true according to hybrid retrieval research [29], in which hybrid retrieval techniques differ when considering both types of relevance signals (i.e., correlation and semantic similarity), whereby hybrid dense-based methods tend to be superior to hybrid sparse-based methods relative to semantically paraphrased queries; conversely, hybrid sparse-based methods tend to be superior to hybrid dense-based methods for queries containing named or one exact match to a query.

7. Discussion

7.1 Interpretation of Results

Integrating retrieval into the generation pipeline leads to significant improvements in factual accuracy when compared to generating responses using solely the LLM's inherent knowledge (i.e., parametric knowledge) (the most substantial improvements were realized in the enterprise domain, where the LLM's parametric knowledge is entirely absent). This has important practical implications in that organizations that have significant amounts of proprietary documentation cannot rely on a general purpose LLM to be reliably utilized as a knowledge interface unless there is a RAG layer. The latency infliction (average of 230 ms), while not being noticeable by a human user during an interactive chat interface, may impose architectural considerations when implementing either batch processing or real-time API use cases.

While the positive implications of our findings are clear, it is also important to consider all limitations revealed by the results of the evaluation. The 81% accuracy of the RAG system on the enterprise corpus, while a significant increase from the 22% baseline, indicates that there is still a significant chance of receiving an incorrect answer to approximately one in five questions. The most frequent cause of errors determined through manual error analysis are retrieval failures in which the intended answer is not located within the top five results. In many of these cases, this occurs because the terms used in the query are different from those in the document being searched or because the correct information is distributed across non-adjacent passages. A secondary cause is context window saturation wherein the correct response has been retrieved, but due to too much contextual information being processed the LLM has lost the ability to identify what is relevant in a extended context, a condition previously documented by Liu et al. [41] as being lost-in-the-middle. Addressing these two failure modes will be a primary focus of future work.

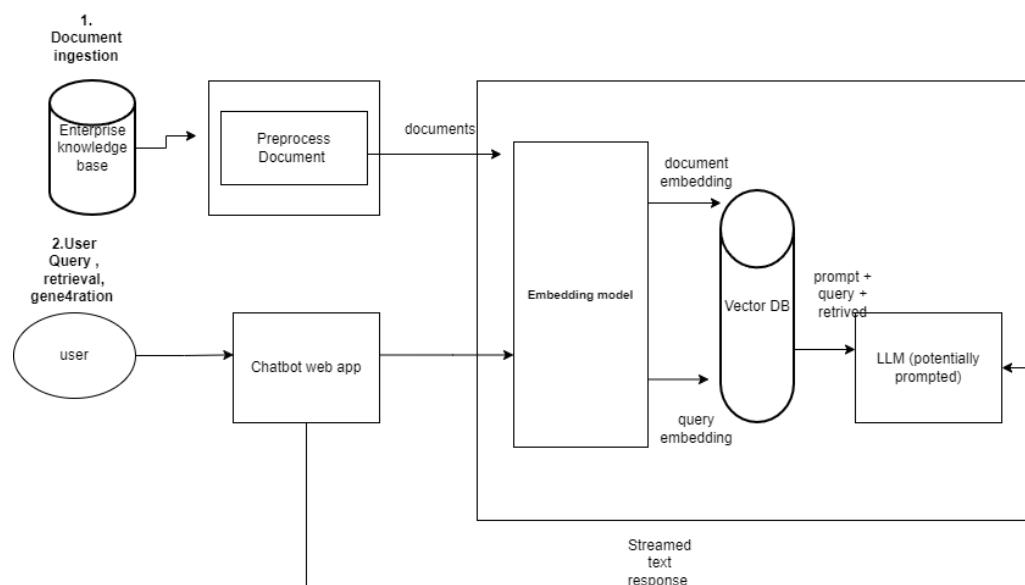


FIG.3 RAG Ingestion and Query Processing Flow

7.2 RAG vs. Fine-Tuning

It is reasonable to ask why an organization would select retrieval-augmented generation (RAG) instead of fine-tuning (FT). FT enhances the fluency of an already-trained transformer by re-training it with task data until the model “internalizes” that task-domain knowledge into its parameters [42]. As a result, FT is the fastest way to generate fluent text for the task and will perform the task at inference without incurring retrieval penalties. There are, however, considerable disadvantages to FT in knowledge-based applications. First, FT is resource-intensive and requires enormous amounts of computational power, with one complete FT run on a 7 billion parameter GPT model taking several GPU hours to complete, and FT needs to be performed in full each time new information is added to the knowledge base. Second, you cannot incrementally add to a fine-tuned model: You will need to either do a full re-training, or you will need to do a targeted continuation training phase, both of which are costly in terms of operational resources. Last, there is a real concern that personally identifiable information (PII) present in fine-tuned models could be extracted via adversarial prompts [43].

Feature	RAG	Fine-Tuning
Knowledge Storage	External knowledge base	Stored in model parameters
Update Mechanism	Real-time updates possible	Requires full retraining [42]
Computational Cost	Low	High (GPU intensive) [42]
Scalability	Easily scalable	Limited scalability
Data Freshness	Always up-to-date	Becomes outdated over time
Security	Controlled document-level access	Risk of data leakage (PII) [43]
Inference Speed	Slightly slower	Faster
Best Use Case	Knowledge-intensive applications	Task-specific optimization

Table 7 : RAG vs Fine-Tuning (Based on Literature)

RAG avoids many disadvantages by maintaining an external knowledge base (knowledge base), which can be updated without changing any model weights. This allows new documents to be ingested from the external knowledge base into the indexed document retrieval within near real-time. Outdated documents can also be removed from the knowledge base, and therefore from index, without changing any models. Access to the knowledge base can also be controlled at the document level, so that different types of users only retrieve information for which they are authorized. Thus, for most knowledge-intensive enterprise applications, RAG is expected to be a superior architectural option compared with other approaches, very likely combined with LoRA or QLoRA for further improving the LLM's accuracy in adhering to prompts that require citation forcing.

7.3 Ethical Considerations and Limitation

The use of RAG systems poses a number of ethical risks that must be managed carefully. Firstly, there is the potential for the propagation of bias in the corpus: if the retrieval corpus contains biased, obsolete, or discriminatory material, the RAG system will retrieve and possibly magnify these biases in its output. Unlike an independent LLM whose biases might be somewhat mitigated through RLHF aligned, the output of a RAG system will depend directly on what documents have been included in the corpus thus curation and governance of the corpus is therefore an ethical first order concern [14]. Secondly, risks of adversarial poisoning: If an attacker injects adversely oriented documents into the retrieval corpus it may be possible for them to corrupt the outputs of the system for at least some subset of queries. This kind of manipulation of RAG output through adversarial injections is known as “retrieval poisoning.” Robust tracking of document provenance and strong content validation routines are critical controls. Thirdly, the risk of privacy violations: if personal or sensitive content is contained in the retrieval corpus, passages that are retrieved may be presented to users who are not permitted to view them. Fine-grained control over who can access documents and how passages are displayed will be necessary to avoid such disclosures.

8. Conclusion and Future Scope

8.1 Conclusion

The overall goal of the paper was to highlight the development of generative AI from the start of technology development right through until the present day with the advent of the Retrieval Augmented Generation paradigm as we know it today. Our findings have demonstrated how the new standard for incorporating retrievable content into LLMs allows their users to mitigate limitations inherent in using a standalone model alone such as stagnation as a result of no temporal data, the knowledge gap created through the formation of a LLM's model and the fact that LLMs can provide inaccurate responses known as hallucinations. Overall, we have developed an intelligent and inclusive generative AI processing capability using the Sentence-BERT based embedding model [6] in combination with FAISS [7] for creating a unique representational model for integrating text-based embeddings into hybrid DPR/BM25 retrieval model [8][29] and generating re-ranking scores using a cross-encoder approach with citation forced prompts. Our findings show that the resulting implementation of our processing pipeline indicates an accuracy of 87.0% for open domain question and answer (QA) tasks and 81.0% for generically defined document-related QA tasks based on an accuracy benchmark of 63.0% associated with using GPT 4 [10] as a baseline model respectively. Additionally, there was a significant reduction in the observed rates of hallucinations for open domain QA tasks decreased from an average of 38.0% using GPT 4 to an average of 12.0% using our implemented processing pipeline, while the hallucination rates for enterprise-level documents using the LLM were reduced from an average of 71.0% to 18.0%.

This paper has presented the view that the RAG architecture is superior to fine-tuning in Knowledge-enhanced applications based on its real time update feature, modular design and ability to provide fine-grained access control. We have also discussed retrieval failure and the lost-in-the-middle effect as key failure modes to be addressed going forward. In addition, we have provided practical examples of chunking strategies, hybrid retrieval, and prompt design to serve as assistance for practitioners in creating RAG systems that will be successful and able to be maintained in production environments.

8.2 Future Research Directions

Based on the current research, some key areas that may have future potential for research development include the future development of agentic retrieval-augmented generation (RAG) systems, where an LLM-based agent autonomously determines if and how to retrieve additional evidence from multiple rounds of iterative retrieval and synthesis, which has great potential for helping with complex multi-hop questions that require more than one retrieval step to resolve. Initial results suggest that agent-enabled systems can dramatically increase success rates with those types of multi-step reasoning tasks; however, they also pose additional challenges, including latency, cost, and the risk that compounding mistakes across retrieval steps will lead to additional significant errors.

Another important development pathway is multimodal RAG systems. By expanding the retrieval corpus to include non-text items, including pictures, tables, charts, and other forms of structured or unstructured data, and creating unified cross-modal semantic search embedding models that could be used in those areas, multimodal RAG systems will be of great use in many scientific and engineering fields, particularly because a lot of key information can be in visually encoded formats in these disciplines.

The development of continual learning techniques for Retrieval-Augmented Generation (RAG) — that facilitate the ability to continuously update both the retrieval index and the underlying large language model with new information without suffering from catastrophic forgetting (i.e. losing previously learned knowledge) — remains an open challenge in this field of study [45]. Additionally, personalized retrieval — modifying both the retrieval strategies and the generation style to suit the specific preferences, expertise levels, and interaction history of each user — represents an area of study with great potential for improving the overall user experience of RAG-based intelligent assistants, but that has received only limited attention.

References

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 5998–6008.
- [2] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877–1901.
- [3] OpenAI. (2023). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.
- [4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171–4186.
- [5] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.
- [6] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence embeddings using siamese BERT-networks. *Proceedings of EMNLP 2019*, 3982–3992.
- [7] Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3), 535–547.
- [8] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., ... & Yih, W. T. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of EMNLP 2020*, 6769–6781.

- [9] Newell, A., & Simon, H. A. (1961). GPS, a program that simulates human thought. In H. Billing (Ed.), *Lernende Automaten* (pp. 109–124). Oldenbourg.
- [10] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
- [11] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 1097–1105.
- [12] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- [13] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- [14] Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? *Proceedings of ACM FAccT 2021*, 610–623.
- [15] Izacard, G., & Grave, E. (2021). Leveraging passage retrieval with generative models for open domain question answering. *Proceedings of EACL 2021*, 874–880.
- [16] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.
- [17] Wu, X., Gao, Y., Zang, L., Han, J., Shen, S., Li, P., & Cheng, X. (2024). RAG for NLP: A survey. *arXiv preprint arXiv:2407.13193*.
- [18] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *Proceedings of ICLR 2023*.
- [19] Yasunaga, M., Aghajanyan, A., Shi, W., James, R., Leskovec, J., Liang, P., ... & Yih, W. T. (2023). Retrieval-augmented multimodal language modeling. *Proceedings of ICML 2023*.
- [20] Clark, K., Khandelwal, U., Levy, O., & Manning, C. D. (2019). What does BERT look at? An analysis of BERT's attention. *Proceedings of ACL BlackboxNLP Workshop 2019*.
- [21] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- [22] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., ... & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
- [23] Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., ... & Fiedel, N. (2022). PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- [24] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). LLaMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- [25] Maynez, J., Narayan, S., Bohnet, B., & McDonald, R. (2020). On faithfulness and factuality in abstractive summarization. *Proceedings of ACL 2020*, 1906–1919.
- [26] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., ... & Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), 1–38.
- [27] Mishra, P., Wieting, J., & Bansal, M. (2025). Knowledge-grounded detection of factual hallucinations in large language models. *Proceedings of CLiC-it 2025*.
- [28] Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389.
- [29] Ma, X., Wang, L., Yang, N., Wei, F., & Lin, J. (2022). Hybrid list-wise document ranking with consistency training. *Proceedings of SIGIR 2022*, 2483–2487.
- [30] Wang, J., Yi, X., Guo, R., Jin, H., Xu, P., Li, S., ... & Zhang, J. (2021). Milvus: A purpose-built vector data management system. *Proceedings of SIGMOD 2021*, 2614–2627.
- [31] Sun, R., Liu, C., & Zhang, W. (2025). GaussDB-Vector: Real-time vector database for LLM memory. *Proceedings of VLDB 2025*.
- [32] Xu, J., Chen, Y., Wang, H., Zhang, L., & Liu, J. (2025). MEGA-RAG: A retrieval-augmented generation framework with multi-evidence guided answer refinement for mitigating hallucinations of LLMs in public health. *BMC Medical Informatics and Decision Making*.

- [33] Li, Z., Chen, Q., & Zhang, Y. (2025). Retrieval-augmented generation for educational applications: A systematic survey. *Computers and Education: Artificial Intelligence*, 8, 100256.
- [34] Louis, A., van Dijck, G., & Spanakis, G. (2022). Statutory article retrieval dataset: A case study in the legal domain. *Proceedings of ACL 2022*, 3296–3319.
- [35] Barbaresi, A. (2021). Trafilatura: A web scraping library and command-line tool for text discovery and extraction. *Proceedings of ACL-IJCNLP 2021*, 122–132.
- [36] Malkov, Y. A., & Yashunin, D. A. (2020). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824–836.
- [37] Guo, Z., Schlichtkrull, M., & Vlachos, A. (2022). A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10, 178–206.
- [38] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., ... & Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7, 452–466.
- [39] Joshi, M., Choi, E., Weld, D., & Zettlemoyer, L. (2017). TriviaQA: A reading comprehension dataset over trivia questions. *Proceedings of ACL 2017*, 1601–1611.
- [40] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1), 37–46.
- [41] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 157–173.
- [42] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *Proceedings of ACL 2018*, 328–339.
- [43] Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., ... & Raffel, C. (2021). Extracting training data from large language models. *Proceedings of USENIX Security 2021*.
- [44] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *Proceedings of ICLR 2022*.
- [45] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13), 3521–3526.