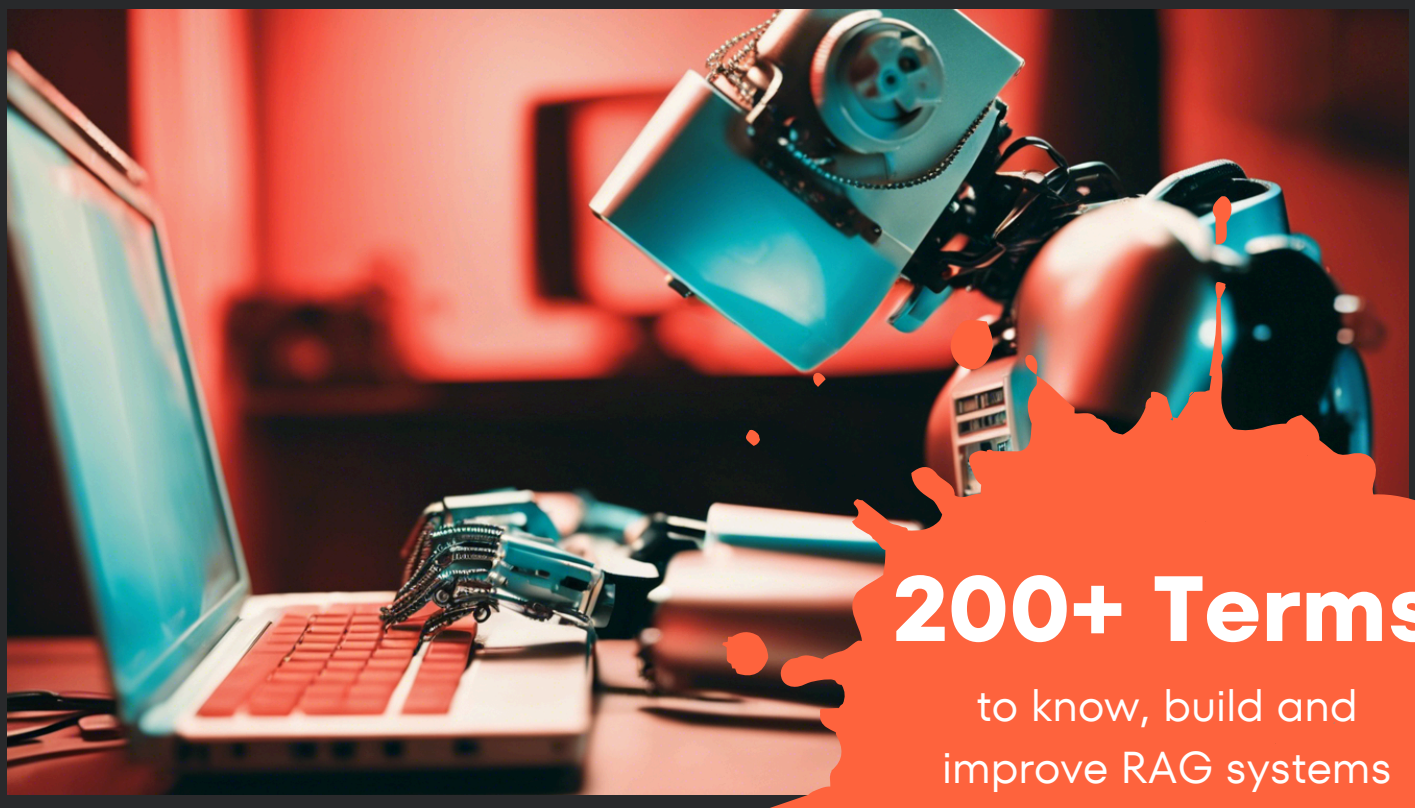
THE RISE OF CONTEXTUAL AI

A Taxonomy of Retrieval Augmented Generation

Components, Concepts, Use Cases & more...

Abhinav Kimothi

October 2024



200+ Terms

to know, build and improve RAG systems

Introduction

Retrieval Augmented Generation, or RAG, stands as a pivotal technique shaping the landscape of the applied generative AI. A novel concept introduced by Lewis et al in their seminal paper Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, RAG has swiftly emerged as a cornerstone, enhancing reliability and trustworthiness in the outputs from Large Language Models (LLMs).

In 2024, RAG is one of the most widely used techniques in generative AI applications and as per Databricks, at least 60% of LLM applications utilise some form of RAG. RAG's acceptance is also propelled by the simplicity of the concept. Simply put, a RAG system searches for information from a knowledge base and sends it along with the query to the LLM for the response.

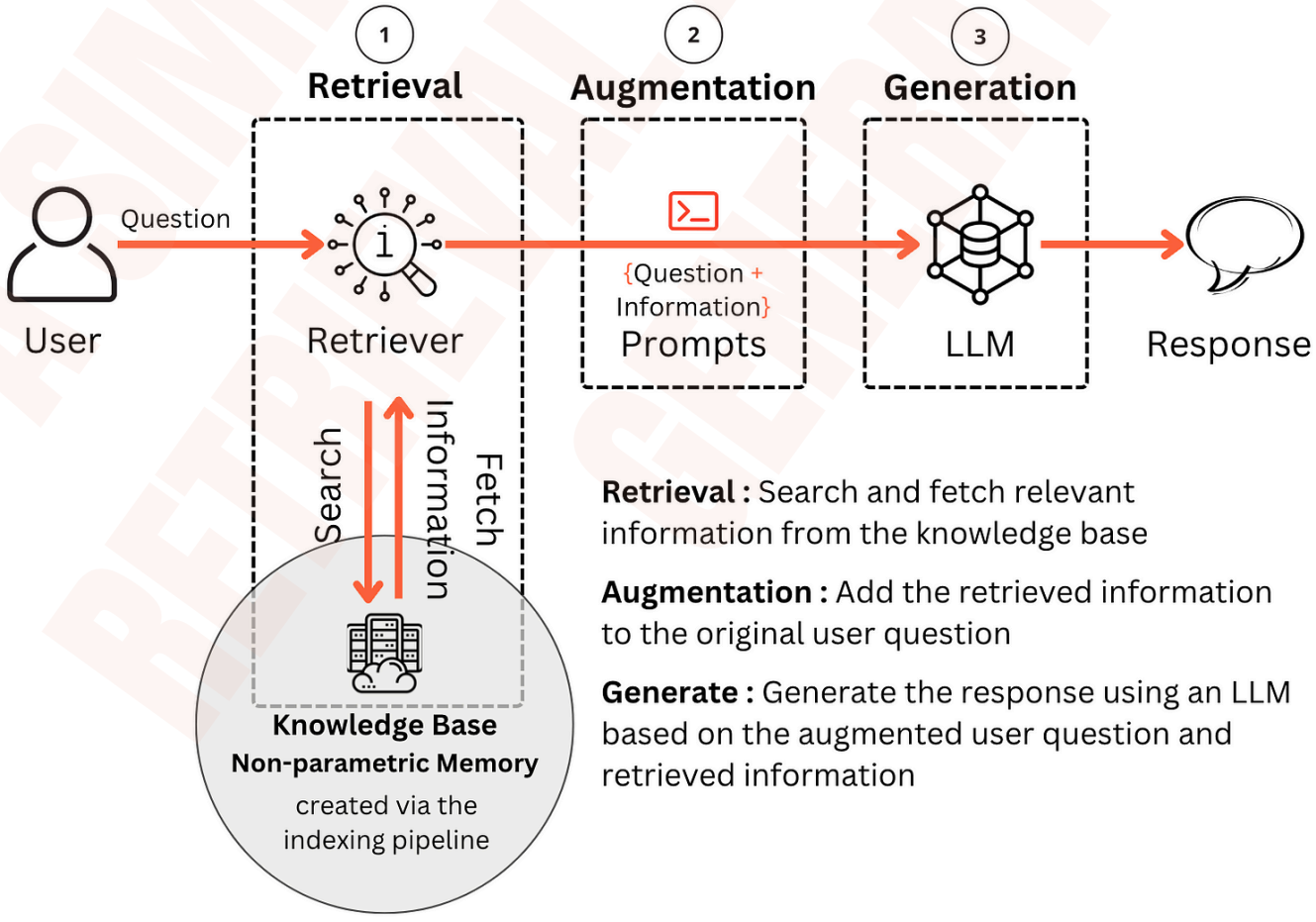


Figure 1: Retrieval Augmented Generation enhances the reliability and the trustworthiness in LLM responses (Source: A Simple Guide to Retrieval Augmented Generations)

Introduction

RAG today encompasses a wide array of techniques, models, and approaches. It can get a little overwhelming for newcomers. As RAG continues to evolve it's crucial to create a shared language framework for researchers, practitioners, developers and business leaders.

This taxonomy is an attempt to clarify the components of RAG, serve as a guide for understanding key building blocks and provide a roadmap to navigate through this, somewhat complex, evolving RAG ecosystem.

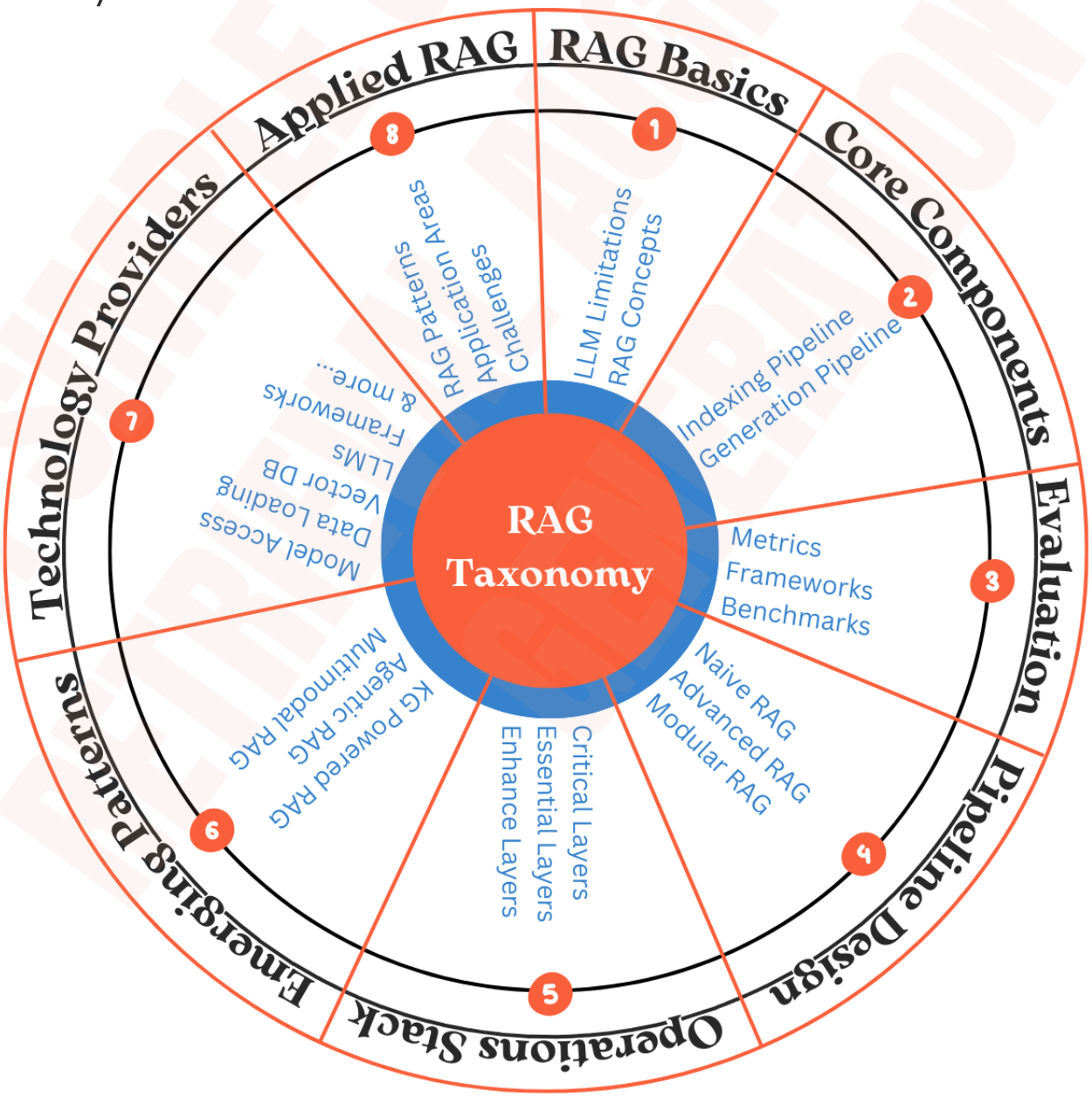
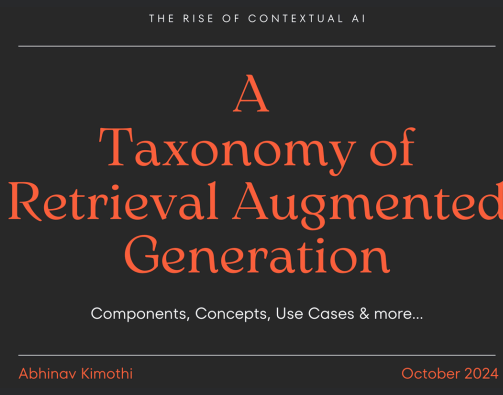
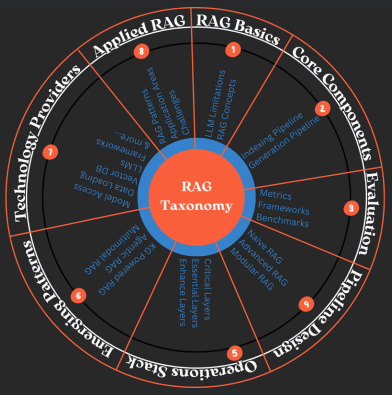


Figure 2: Themes for A Taxonomy of Retrieval Augmented Generations

A DOWNLOADABLE PDF VERSION OF THIS EBOOK IS ALSO AVAILABLE ON GUMROAD





SCAN CODE OR CLICK HERE TO DOWNLOAD

Table of Contents

LLM Limitations

Knowledge Cut-off Date: Training an LLM is an expensive and time-consuming process. It takes massive volumes of data and several weeks, or even months, to train an LLM. The data that LLMs are trained on is therefore not current. For example, GPT-4o has knowledge only up to October 2023. Any event that happened after this knowledge cut-off date is not available to the model.

Training Data Limitation: LLMs are trained on large volumes of data from a variety of public sources—like Llama 3 has been trained on a whopping 15 trillion tokens (about 7 times more than Llama 2)—but they do not have any knowledge of information that is not public. Publicly available LLMs have not been trained on information like internal company documents, customer information, product documents, etc. So, LLMs cannot be expected to respond to queries about such information.

Hallucinations: LLMs are next-word predictors. They are not trained to verify the facts in their responses. Thus, it is observed that LLMs sometimes provide responses that are factually incorrect, and despite being incorrect, these responses sound extremely confident and legitimate. This characteristic of "lying with confidence," called hallucination, has proved to be one of the biggest criticisms of LLMs.

Context Window: Every LLM, by the nature of the architecture, can process upto a maximum number of tokens. This maximum number of tokens is referred to as the context window of the model. If the prompt is longer than the context window, then the portion of the prompt beyond the limit is discarded.

Model Name	Developer	Knowledge Cut-off Date	Context Window (Tokens)
Claude 3	Anthropic	March 2024	200k tokens
GPT-4o	OpenAI	April 2023	128k tokens
LLaMA 3.1	Meta	June 2024	128k tokens
PaLM 2	Google DeepMind	April 2023	32k tokens
Gemini 1.5 Pro	Google DeepMind	Early 2024	256k tokens
Claude 2	Anthropic	Early 2023	100k tokens
Mistral	Mistral AI	2023	8k tokens
Falcon 40B	TII	March 2023	2k tokens
BLOOM	BigScience	Early 2022	2k tokens
GPT-NeoX-20B	EleutherAI	April 2022	2k tokens

Table 1: Popular LLMs with their cut-off date and context window

RAG Concepts

Parametric Memory: The ability of an LLM to retain information that it has been trained on is solely reliant on its parameters. It can therefore be said that LLMs store factual information in their parameters. This memory that is internally present in the LLM can be referred to as the parametric memory. This parametric memory is limited. It depends upon the number of parameters and is a factor of the data on which the LLM has been trained on.

Non-parametric Memory: Information that LLMs do not have in their internal parameters but can access via external retrieval mechanisms, like a search engine or database. RAG provides the LLM with access to this non-parametric memory.

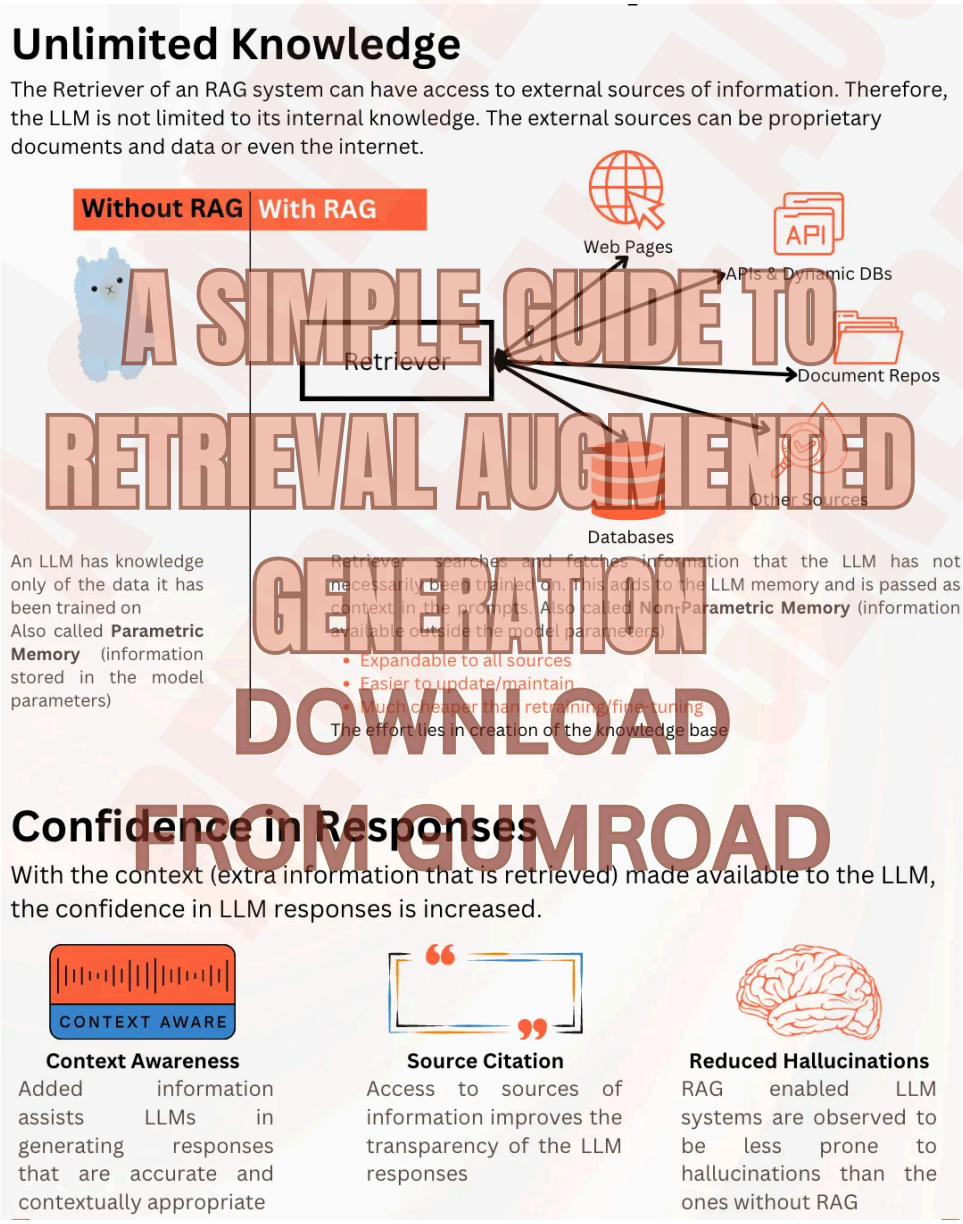
Knowledge Base: The non-parametric memory that has been created for the RAG application. This contains information from a variety of pre-determined sources which is processed and stored in persistent memory

User Query: The prompt that a user (or a system) wants to send to an LLM for a response

Retrieval: The process via which, information pertinent to the user query is searched for and fetched from the knowledge base.

Augmentation: The process of adding the retrieved information to the user query.

Generation: The process of generating results by the LLM when provided with an augmented prompt.



Source Citation: Ability of a RAG system to point out to the information from the knowledge base that was used to generate the response

Unlimited Memory: The notion that any number of documents can be added to the RAG knowledge base

Figure 3: How does RAG help?

Indexing Pipeline: The set of processes that is employed to create the knowledge base for RAG applications. It is a non real-time pipeline that updates the knowledge base at periodic intervals.

Source Systems: The original locations where the data that is desired for the RAG application is stored. These can be data lakes, file systems, CMSs, SQL & NoSQL databases, 3rd party data stores etc.

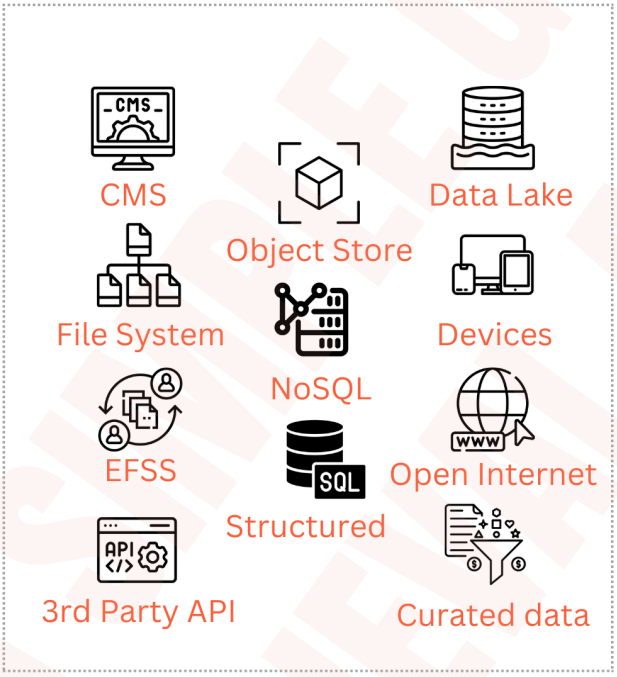


Figure 4: Examples of source systems that can be connected to

Data Loading: The first step of the indexing pipeline that connects to source systems to extract and parse files for data to be used in the RAG knowledge base.

Metadata: A common way of defining metadata is "data about data". Metadata describes other data. It can provide information like a description of the data, time of creation, author, etc. While metadata is useful for managing and organising data, in the context of RAG, metadata enhances the search-ability of data.

Data Masking: Obfuscation of sensitive information like PII or confidential data

Chunking: The process of breaking down long pieces of text into smaller manageable sizes or "chunks". Chunking is crucial to the efficient creation of knowledge base for RAG systems. Chunking increases the ease of search and overcomes the context window limits of LLMs.

Lost in the middle problem: Even in those LLMs which have a long context window (Claude 3 by Anthropic has a context window of up to 200,00 tokens), an issue with accurately reading the information has been observed. It has been noticed that accuracy declines dramatically if the relevant information is somewhere in the middle of the prompt. This problem can be addressed by passing only the relevant information to the LLM instead of the entire document.

Fixed Size Chunking: A very common approach is to pre-determine the size of the chunk and the amount of overlap between the chunks. There are several chunking methods that follow a fixed size chunking approach. Chunks are created based on a fixed number of characters, tokens, sentences or paragraphs.

Structure-Based Chunking: The aim of chunking is to keep meaningful data together. If we are dealing with data in form of HTML, Markdown, JSON or even computer code, it makes more sense to split the data based on the structure rather than a fixed size.

Context-Enriched Chunking: This method adds the summary of the larger document to each chunk to enrich the context of the smaller chunk. This makes more context available to the LLM without adding too much noise. It also improves the retrieval accuracy and maintains semantic coherence across chunks. This is particularly useful in scenarios where a more holistic view of the information is crucial. While this approach enhances the understanding of the broader context, it adds a level of complexity and comes at the cost of higher computational requirements, increased storage needs and possible latency in retrieval.

Agentic Chunking: In agentic chunking, chunks from the text are created based on a goal or a task. Consider an e-commerce platform wanting to analyse customer reviews. The best way for the reviews to be chunked is if the reviews pertaining to a particular topic are put in the same chunk. Similarly, the critical reviews and positive reviews may be put in different chunks. To achieve this kind of chunking, we will need to do sentiment analysis, entity extraction and some kind of clustering. This can be achieved by a multi-agent system. Agentic chunking is still an active area of research and improvement.

Semantic Chunking: This method looks at semantic similarity (or similarity in the meaning) between sentences is called semantic chunking. It first creates groups of three sentences and then merges groups that are similar in meaning. To find out the similarity in meaning, this method uses Embeddings. This is still an experimental chunking technique.

Small to big chunking: A hierarchical chunking method where the text is first broken down into very small units (e.g., sentences, paragraphs), and the small chunks are merged into larger ones until the chunk size is achieved. Sliding window chunking uses overlap between chunks to maintain context across chunk boundaries.

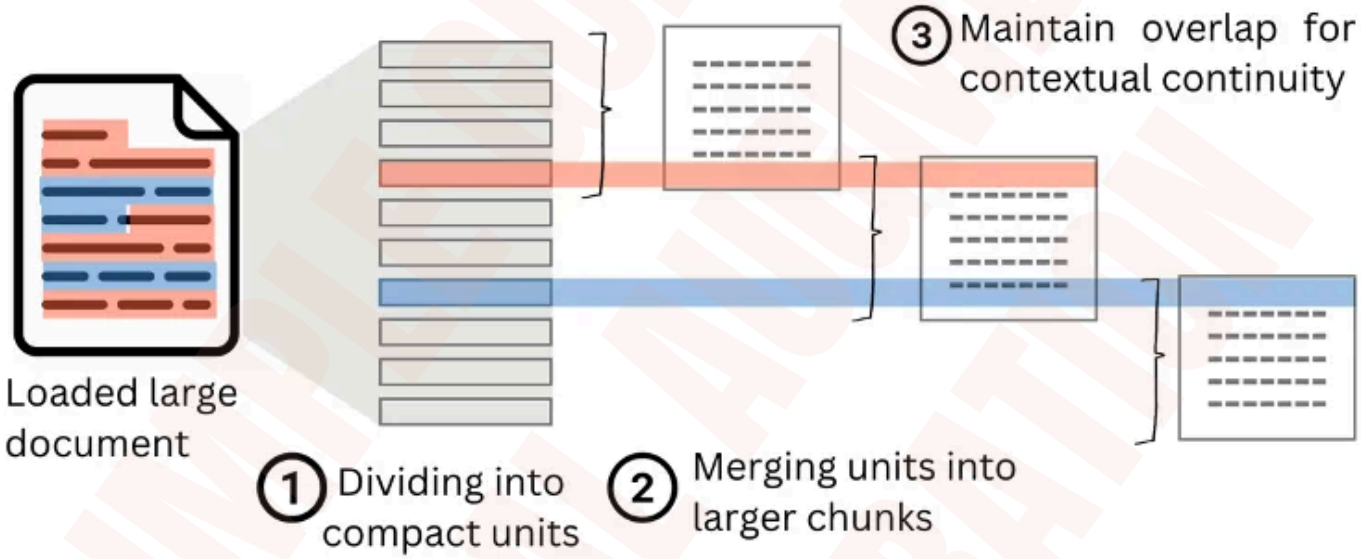


Figure 5: Big to Small Sliding Window Chunking

Chunk Size: The size of the chunks, which is measure in the number of tokens in the chunk, can have a significant impact on the quality of the RAG system. While large sized chunks provide better context, they also carry a lot of noise. Smaller chunks, on the other hand, have precise information but they might miss important information.

Metadata Filtering: Adding metadata like timestamp, author, category, etc. can enhance the chunks. While retrieving, chunks can first be filtered by relevant metadata information before doing a similarity search. This improves retrieval efficiency and reduces noise in the system. For example, using the timestamp filters can help avoid outdated information in the knowledge base.

Metadata Enhancement: Metadata like chunk summary, sentiment, category etc. that can be inferred beyond tags like source, timestamp, author etc. can be used to enhance retrieval.

Parent Child Indexing: A document structure where documents are organised hierarchically. The parent document contains overarching themes or summaries, while child documents delve into specific details. During retrieval, the system can first locate the most relevant child documents and then refer to the parent documents for additional context if needed. This approach enhances the precision of retrieval while maintaining the broader context. At the same time, this hierarchical structure can present challenges in terms of memory requirements and computational load.

Embeddings: Computers, at the very core, do mathematical calculations. Mathematical calculations are done on numbers. Therefore, for a computer to process any kind of non-numeric data like text or image, it must be first converted into a numerical form. Embeddings is a design pattern that is extremely useful for RAG. Embeddings are vector representations of data. As a general definition, embeddings are data that has been transformed into n-dimensional matrices. A word embedding is a vector representation of words.

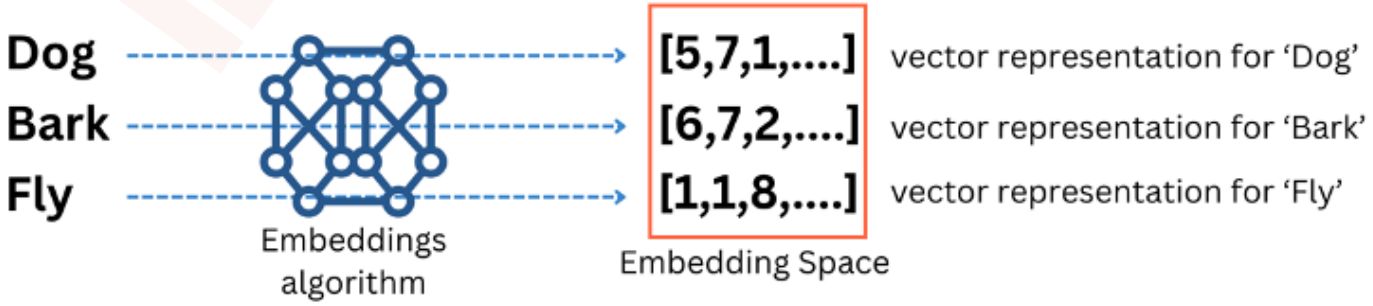


Figure 6: Word Embeddings are vector representation of words

Cosine Similarity: The reason why embeddings are popular is because they help in establishing semantic relationship between words, phrases, and documents. Cosine similarity is calculated as the cosine value of the angle between the two vectors. Cosine of parallel lines i.e. angle=0 is 1 and cosine of a right angle i.e. 90 is 0. On the other end, the cosine of opposite lines i.e. angle =180 is -1. Therefore, the cosine similarity lies between -1 and 1 where unrelated terms have a value close to 0, and related terms have a value close to 1.

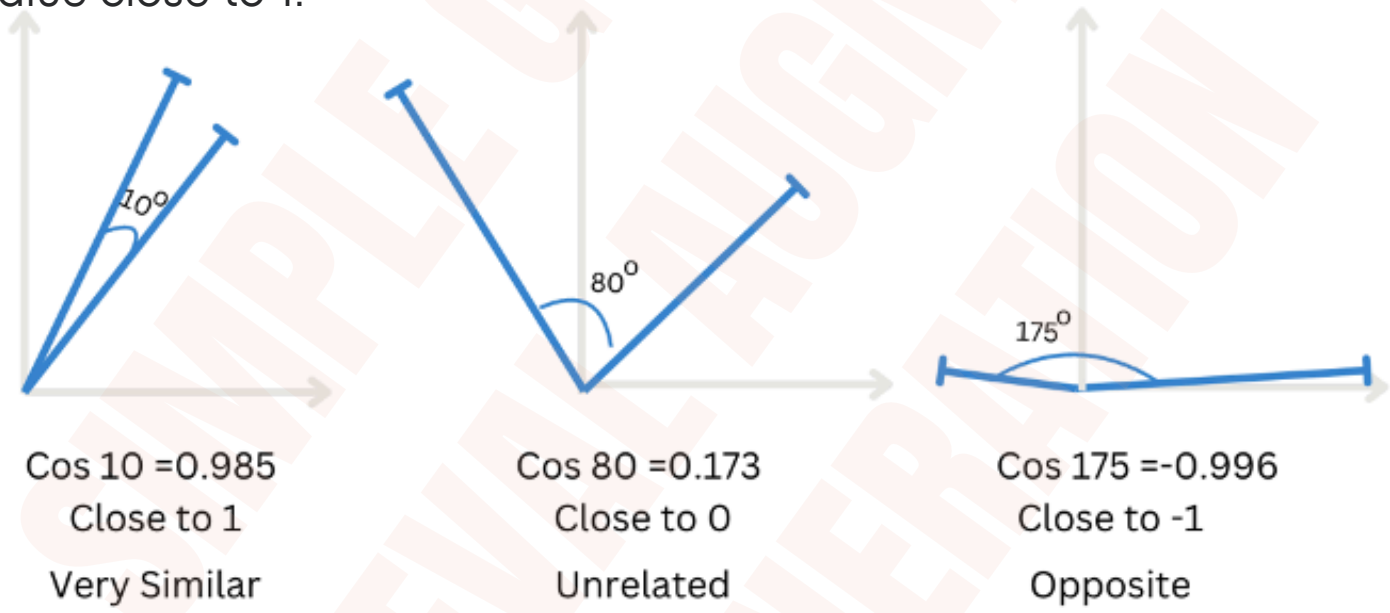


Figure 7: Cosine Similarity between vectors

Word2Vec: Word2Vec is a shallow neural network-based model for learning word embeddings developed by researchers at Google. It is one of the earliest embedding techniques

GloVe: Global Vectors for Word Representations is an unsupervised learning technique developed by researchers at Stanford University

FastText: FastText is an extension of Word2Vec developed by Facebook AI Research. It is particularly useful for handling misspellings and rare words.

ELMo: Embeddings from Language Models was developed by researchers at Allen Institute for AI. ELMo embeddings have been shown to improve performance on question answering and sentiment analysis tasks.

BERT: Bidirectional Encoder Representations from Transformers, developed by researchers at Google, is a Transformers architecture-based model. It provides contextualized word embeddings by considering bidirectional context, achieving state-of-the-art performance on various natural language processing tasks.

Pre-trained Embeddings Models: Embeddings models that have been trained on a large corpus of data can also generalise well across a number of tasks and domains. There are a variety of proprietary and open-source pre-trained embeddings models that are available to use. This is also one of the reasons why the usage of embeddings has exploded in popularity across machine learning applications.

Vector Databases: Vector Databases are built to handle high dimensional vectors like embeddings. These databases specialise in indexing and storing vector embeddings for fast semantic search and retrieval.

Vector Indices: These are libraries that focus on the core features of indexing and search. They do not support data management, query processing, interfaces etc. They can be considered a bare bones vector database. Examples of vector indices are Facebook AI Similarity Search (FAISS), Non-Metric Space Library (NMSLIB), Approximate Nearest Neighbors Oh Yeah (ANNOY), etc

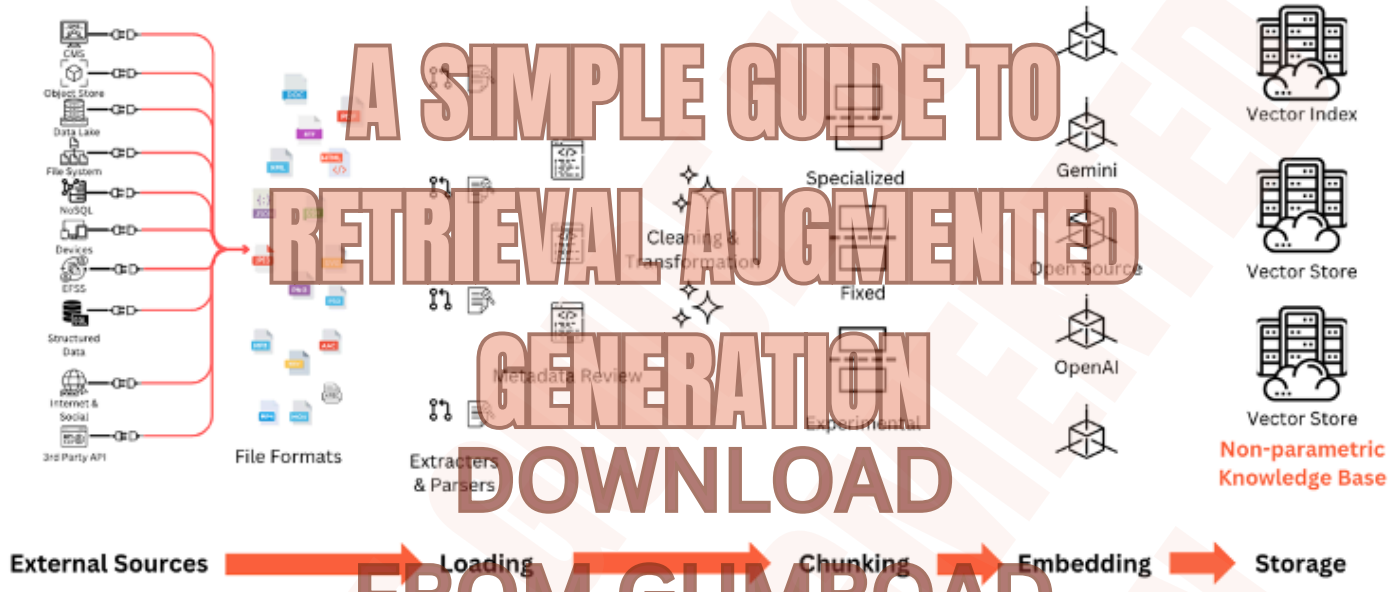


Figure 8: Creation and maintenance of non-parametric database via the indexing pipeline

Generation Pipeline: The set of processes that is employed to search and retrieve information from the knowledge base to generate responses to user queries. It facilitates real-time interaction with users.

Information Retrieval: IR is the science of searching. Whether you are searching for information in a document, or searching for document themselves, it falls under the gamut of Information Retrieval.

Retriever: The component of the generation pipeline that uses an algorithm to search and retrieve relevant information from the knowledge base.

Boolean retrieval: A simple keyword-based search where Boolean logic is used to match documents with queries based on absence or presence of the words. Documents are retrieved if they contain the exact terms in the query, often combined with AND, NOT and OR operators

Core Components

TF-IDF: Term Frequency-Inverse Document Frequency is a statistical measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus). It assigns higher weights to words that appear frequently in a document but infrequently across the corpus.

BM25: Best Match 25 is an advanced probabilistic model used to rank documents based on the query terms appearing in each document. It is part of the family of probabilistic information retrieval models and is considered an advancement over the classic TF-IDF model. The improvement that BM25 brings is that it adjusts for the length of the documents so that longer documents do not unfairly get higher scores.

Static Word Embeddings: Static embeddings like Word2Vec and GloVe represent words as dense vectors in a continuous vector space, capturing semantic relationships based on context. For instance, "king" - "man" + "woman" approximates "queen". These embeddings can capture nuances like similarity and analogy, which BoW, TF-IDF and BM25 miss.

Contextual Embeddings: Contextual embeddings, generated by models such as BERT or OpenAI's text embeddings, produce high-dimensional, context-aware representations for queries and documents. These models, based on transformers, capture deep semantic meanings and relationships. For example, a query about "apple" will retrieve documents discussing apple the fruit or apple the technology company depending on the input query.

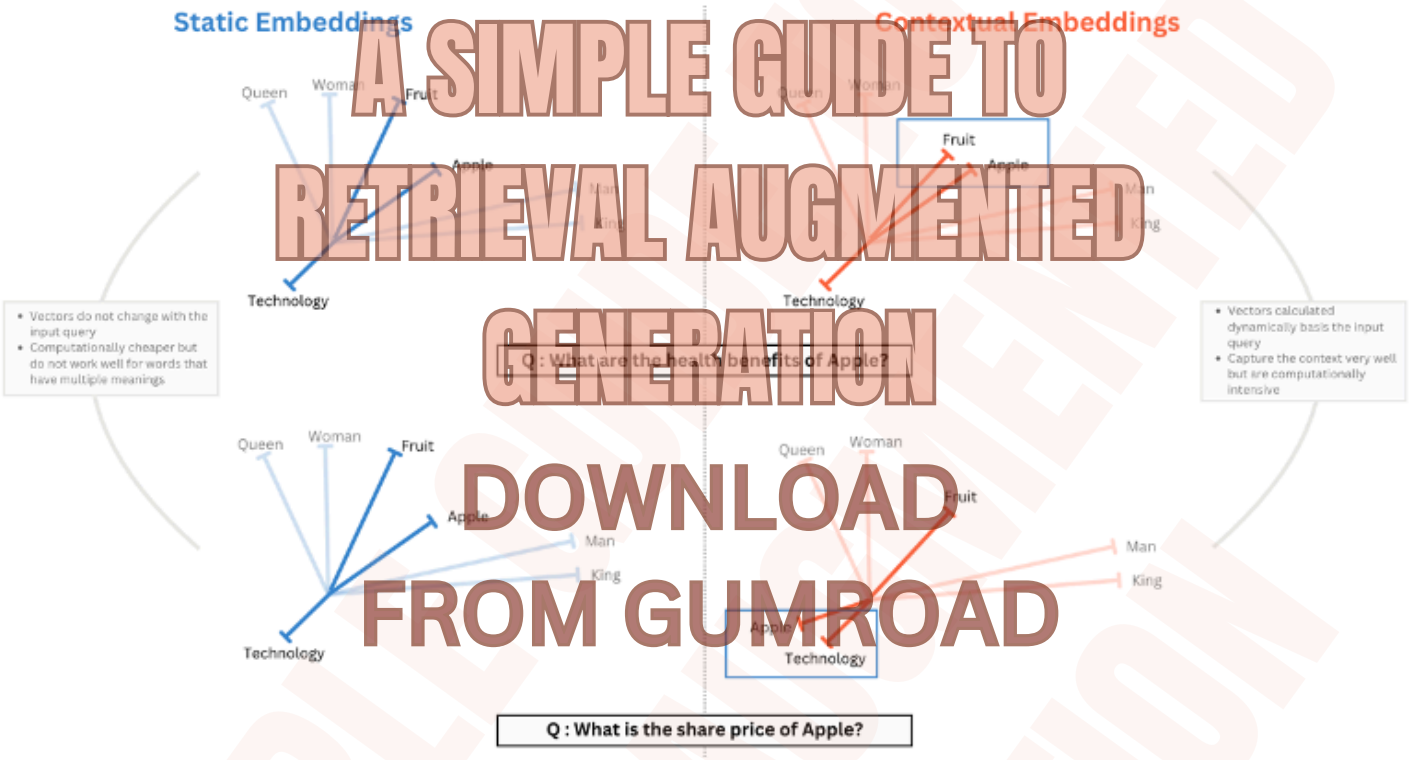


Figure 9: Static Embeddings vs Contextual Embeddings.

Learned Sparse Retrieval: Generate sparse, interpretable representations using neural networks. Examples: SPLADE, DeepCT, DocT5Query

Dense Retrieval: Encode queries and documents as dense vectors for semantic matching. Examples: DPR (Dense Passage Retriever), ANCE, RepBERT

Hybrid Retrieval: Combine sparse and dense methods for balanced efficiency and effectiveness. Examples: ColBERT, COIL

Cross-Encoder Retrieval: Directly compare query-document pairs using transformer models. Example: BERT-based rerankers

Graph-based Retrieval: Leverage graph structures to model relationships between documents. Examples: TextGraphs, Graph Neural Networks for IR

Quantum-inspired Retrieval: Apply quantum computing principles to information retrieval. Example: Quantum Language Models (QLM)

Neural IR models: Encompass various neural network-based approaches to information retrieval. Examples: NPRF (Neural PRF), KNRM (Kernel-based Neural Ranking Model)

Augmentation: The process of combining user query and the retrieved documents from the knowledge base.

Prompt Engineering: The technique of giving instructions to an LLM to attain a desired outcome. The goal of Prompt Engineering is to construct the prompts to achieve accuracy and relevance in the LLM responses with respect to the desired outcome(s).

Contextual Prompting: Adding an instruction like "Answer only based on the context provided." to have set up the generation to focus only on the provided information and not from LLM's internal knowledge (or parametric knowledge).

Controlled Generation Prompting: Adding an instruction like, "If the question cannot be answered based on the provided context, say I don't know." to ensure that the model's responses are grounded in the retrieved information

Few Shot Prompting: LLMs adhere quite well to examples provided in the prompt. While providing the retrieved information in the prompt, specifying certain examples to help guide the generation in the way the retrieved information needs to be used.

Chain of Thought Prompting: It has been observed that the introduction of intermediate "reasoning" steps, improves the performance of LLMs in tasks that require complex reasoning like arithmetic, common sense, and symbolic reasoning.

Self Consistency: While CoT uses a single Reasoning Chain in Chain of Thought prompting, Self-Consistency aims to sample multiple diverse reasoning paths and use their respective generations to arrive at the most consistent answer

Generated Knowledge Prompting: This technique explores the idea of prompt-based knowledge generation by dynamically constructing relevant knowledge chains, leveraging models' latent knowledge to strengthen reasoning.

Tree of Thoughts Prompting: This technique maintains an explorable tree structure of coherent intermediate thought steps aimed at solving problems.

Automatic Reasoning and Tool-use (ART): ART framework automatically interleaves model generations with tool use for complex reasoning tasks. ART leverages demonstrations to decompose problems and integrate tools without task-specific scripting.

Automatic Prompt Engineer (APE): The APE framework automatically generates and selects optimal instructions to guide models. It leverages a large language model to synthesise candidate prompt solutions for a task based on output demonstrations.

Active Prompt: Active-Prompt improves Chain-of-thought methods by dynamically adapting Language Models to task-specific prompts through a process involving query, uncertainty analysis, human annotation, and enhanced inference.

ReAct Prompting: ReAct integrates LLMs for concurrent reasoning traces and task-specific actions, improving performance by interacting with external tools for information retrieval. When combined with CoT, it optimally utilises internal knowledge and external information, enhancing interpretability and trustworthiness of LLMs.

Recursive Prompting: Recursive prompting breaks down complex problems into sub-problems, solving them sequentially using prompts. This method aids compositional generalisation in tasks like math problems or question answering, with the model building on solutions from previous steps.

Foundation Models: Pre-trained large language models that have been trained on massive data. Some LLM developers make the model weights public for fostering collaboration and community driven innovation while others have kept the models closed offering support, managed services and better user experience

Supervised Fine-Tuning (SFT): Supervised fine-tuning is a process used to adapt a pre-trained language model for specific tasks or behaviours like question-answering or chat. It involves further training a pre-trained foundation model on a labeled dataset, where the model learns to map inputs to specific desired outputs.

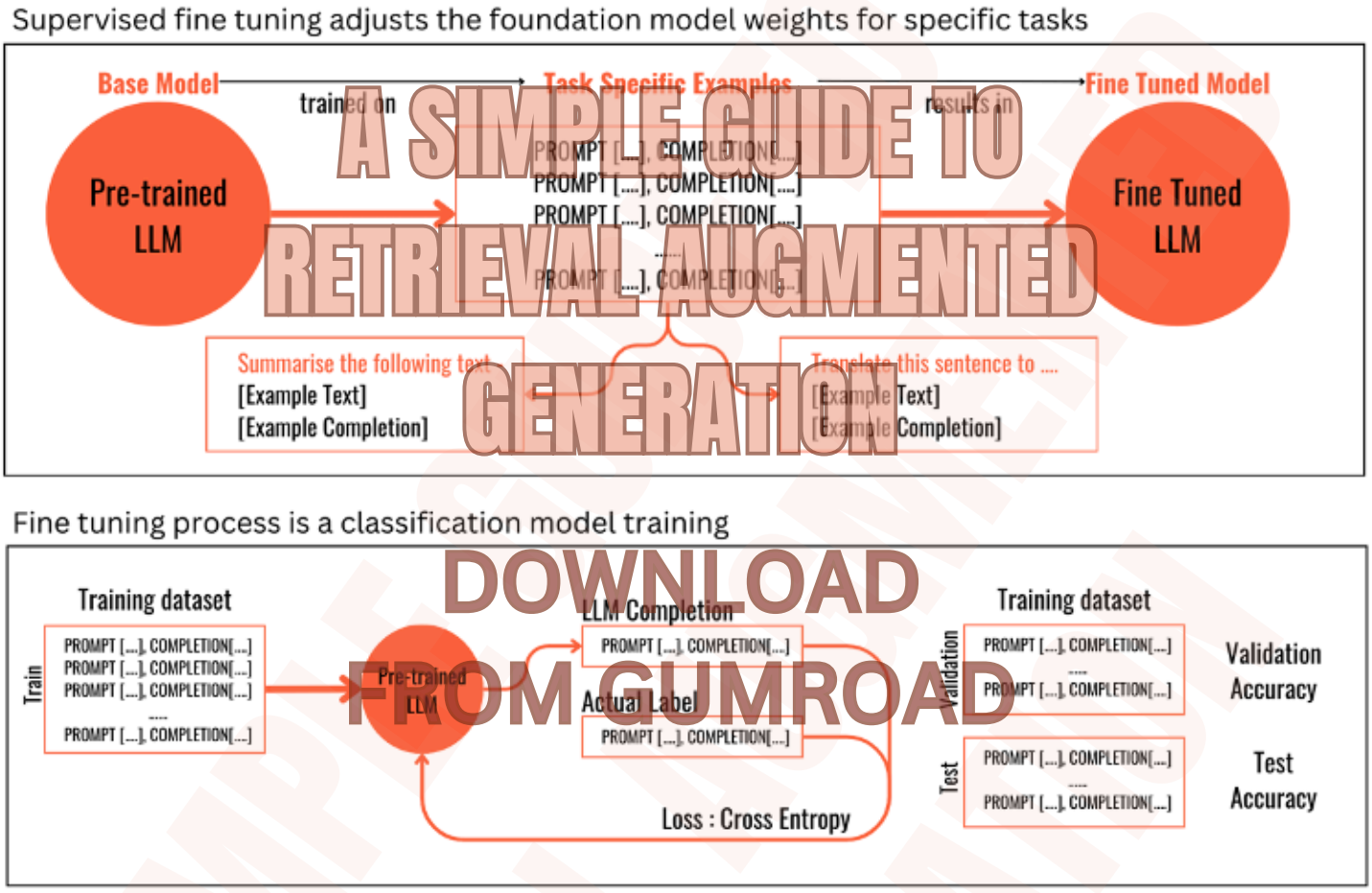
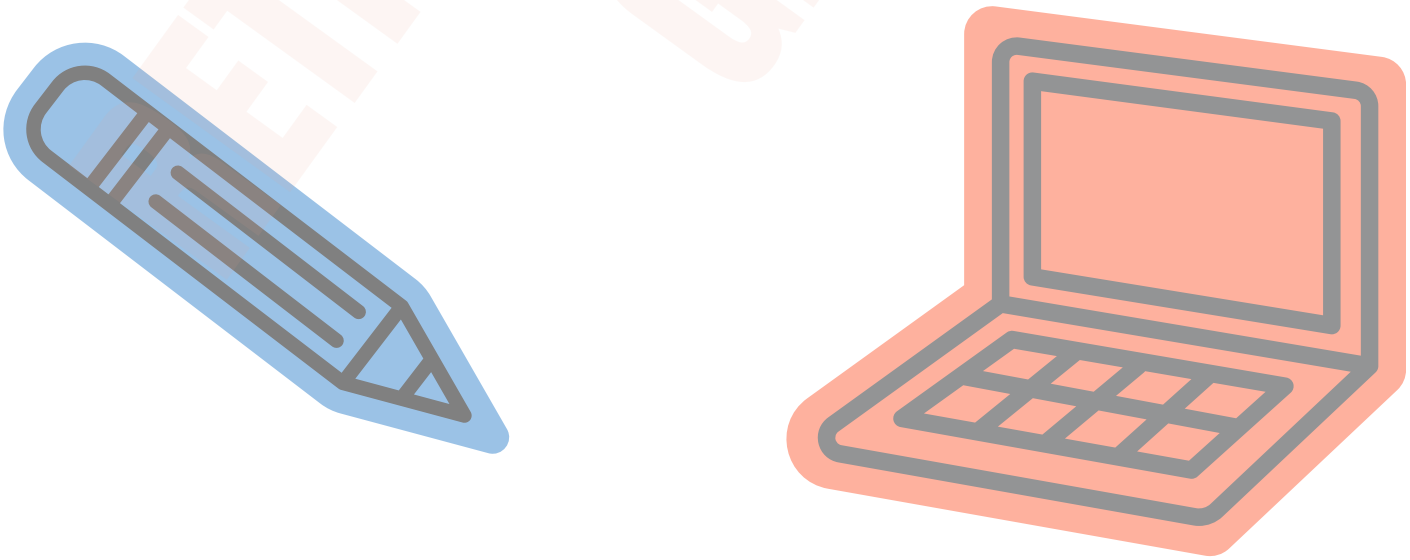


Figure 10: Supervised fine tuning is a classification model training process

Small Language Models (SLMs): Smaller models with parameter sizes in millions or a few billion offer benefits such as faster inference times, lower resource usage and easier deployment on edge devices or resource constrained environments



Evaluation Metrics: Quantitative measures used to evaluate the performance of retrieval & generation and overall RAG system.

Accuracy: The proportion of correct predictions (both true positives and true negatives) among the total number of cases examined. Even though accuracy is a simple, intuitive metric, it is not the primary metric for retrieval. In a large knowledge base, majority of documents are usually irrelevant to any given query, which can lead to misleadingly high accuracy scores. It does not consider ranking of the retrieved results.

Precision: It measures the proportion of retrieved documents that are relevant to the user query. It answers the question, "Of all the documents that were retrieved, how many were actually relevant?"

Precision@k: A variation of precision that measures the proportion of relevant documents amongst the top 'k' retrieved results. It is particularly important because it focusses on the top results rather than all the retrieved documents.

Recall: It measures the proportion of the relevant documents retrieved from all the relevant documents in the corpus. It answers the question, "Of all the relevant documents, how many were actually retrieved?". Like precision, recall also doesn't consider the ranking of the retrieved documents. It can also be misleading as retrieving all documents in the knowledge base will result in a perfect recall value.

F1-score: The harmonic mean of precision and recall. It provides a single metric that balances both the quality and coverage of the retriever.

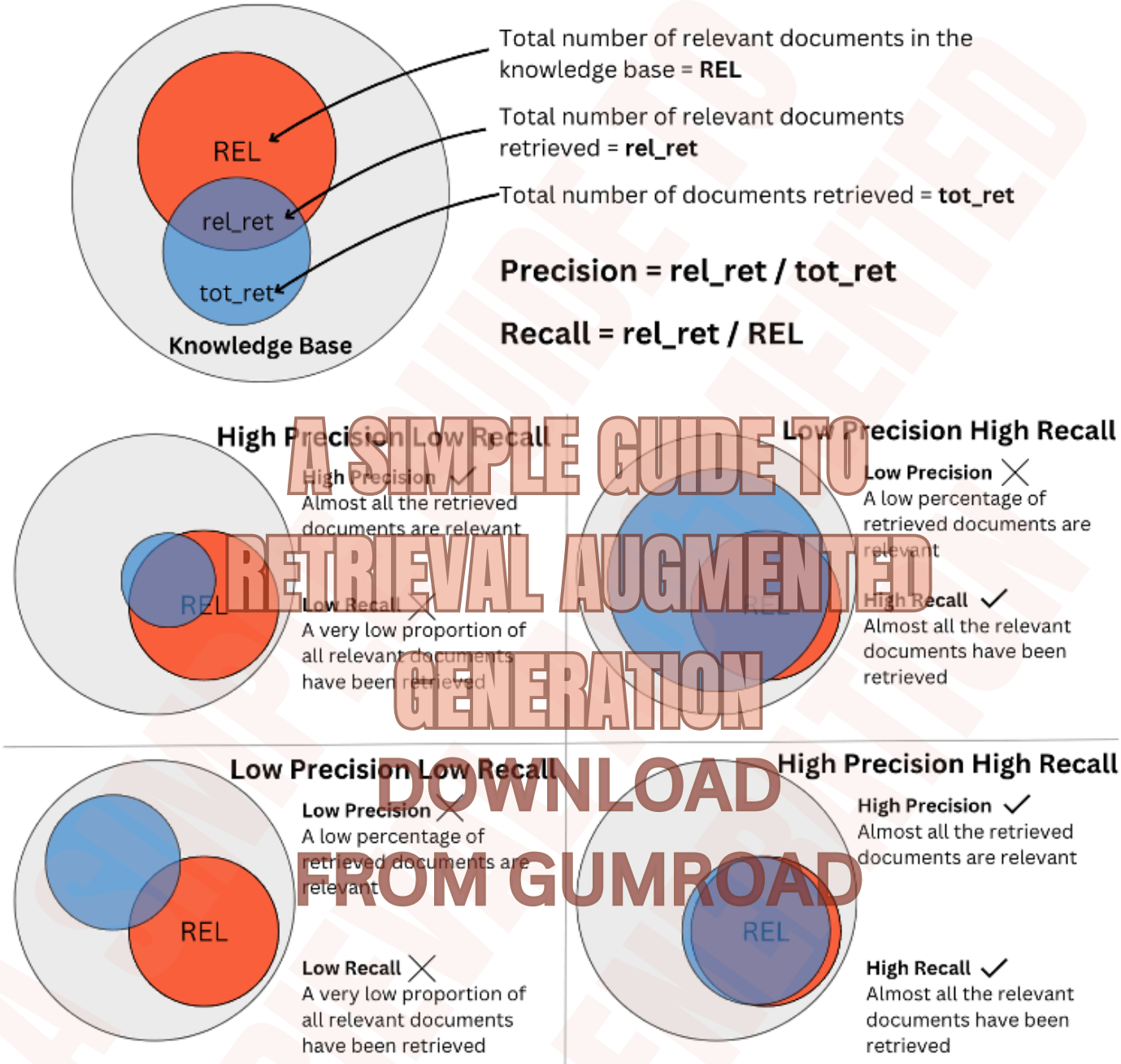


Figure 11: Precision and Recall

Mean Reciprocal Rank (MRR): Useful in evaluating the rank of the relevant document, it measures the reciprocal of the ranks of the first relevant document in the list of results. MRR is calculated over a set of queries.

Mean Average Precision(MAP): A metric that combines precision and recall at different cut-off levels of 'k' i.e. the cut-off number for the top results. It calculates a measure called Average Precision and then averages it across all queries.

Normalised Discounted Cumulative Gain (nDCG): Evaluates the ranking quality by considering the position of relevant documents in the result list and assigning higher scores to relevant documents appearing earlier. It is particularly effective for scenarios where documents have varying degrees of relevance.

Context relevance: Evaluates how well the retrieved documents relate to the original query. The key aspects are topical alignment, information usefulness and redundancy. The retrieved context should contain information only relevant to the query or the prompt. For context relevance, a metric 'S' is estimated. 'S' is the number of sentences in the retrieved context that are relevant for responding to the query or the prompt.

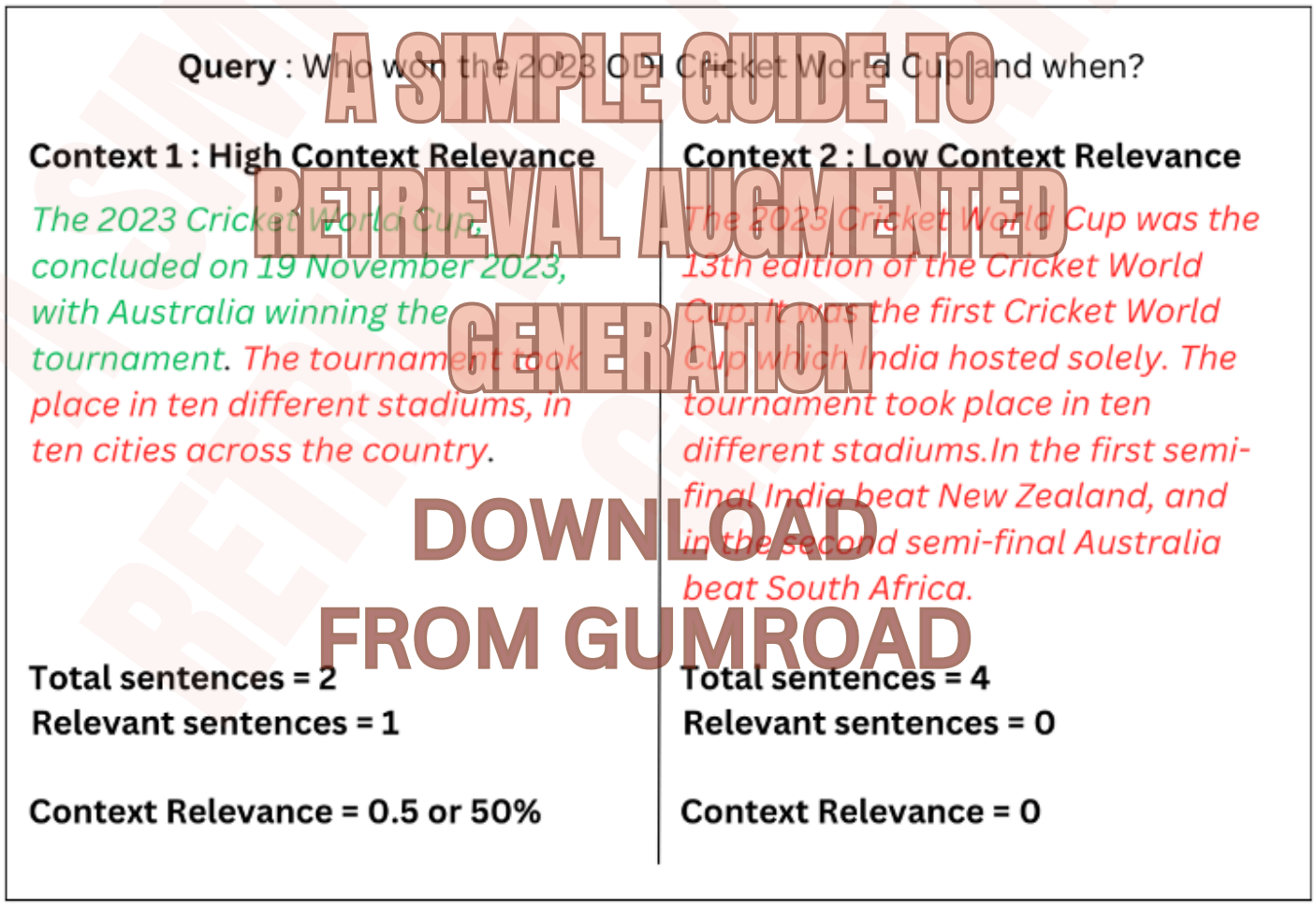


Figure 12: Context relevance evaluates the degree to which the retrieved information is relevant to the query

Answer Faithfulness: A measure of the extent to which the response is factually grounded in the retrieved context. Faithfulness ensures that the facts in the response do not contradict the context and can be traced back to the source. It also ensures that the LLM is not hallucinating. Faithfulness first identifies the number of "claims" made in the response and calculates the proportion of those "claims" present in the context.

Hallucination Rate: Calculate the proportion of generated claims in the response that are not present in the retrieved context

Coverage: Measures the number of relevant claims in the context and calculates the proportion of relevant claims present in the generated response. This measures how much of the relevant information from the retrieved passages is included in the generated answer.

Answer Relevance: Measure of the extent to which the response is relevant to the query. This metric focusses on key aspects such as system's ability to comprehend the query, response being pertinent to the query and the completeness of the response.

Ground truth: Information that is known to be real or true. In RAG, and Generative AI domain in general, Ground Truth is a prepared set of Prompt-Context-Response or Question-Context-Response example, akin to labelled data in Supervised Machine Learning parlance. Ground truth data that is created for your knowledge base can be used for evaluation of your RAG system.

Human Evaluation: A subject matter expert looking at the documents and determines the relevance and accuracy of the outputs.

Noise Robustness: It is impractical to assume that the information stored in the knowledge base for RAG systems is perfectly curated to answer the questions that can be potentially asked of the system. It is very probable that a document is related to the user query but does not have any meaningful information to answer the query. The ability of the RAG system to separate these noisy documents from the relevant ones is Noise Robustness.

Negative Rejection: By nature, Large Language Models always generate text. It is possible that there is absolutely no information about the user query in the documents in the knowledge base. The ability of the RAG system to not give an answer when there is no relevant information is Negative Rejection.

Information Integration: It is also very likely that to answer a user query comprehensively the information must be retrieved from multiple documents. This ability of the system to assimilate information from multiple documents is Information Integration.

Counterfactual Robustness: Sometimes the information in the knowledge base might itself be inaccurate. A high-quality RAG system should be able to address this and reject known inaccuracies in the retrieved information. This ability is Counterfactual Robustness

Frameworks: Tools designed to facilitate evaluation offering automation of the evaluation process and data generation. They are used to streamline the evaluation process by providing a structured environment for testing different aspects of a RAG systems. They are flexible and can be adapted to different datasets and metrics.

RAGAS: Retrieval Augmented Generation Assessment or RAGAs is a framework developed by Exploding Gradients that assesses the retrieval and generation components of RAG systems without relying on extensive human annotations. RAGAs also helps in synthetically generating a test dataset that can be used to evaluate a RAG pipeline

Synthetic Test Dataset Generation: Using models like LLMs to automatically generate ground truth data from the knowledge base.

LLM as a judge: Using an LLM to evaluate a task.

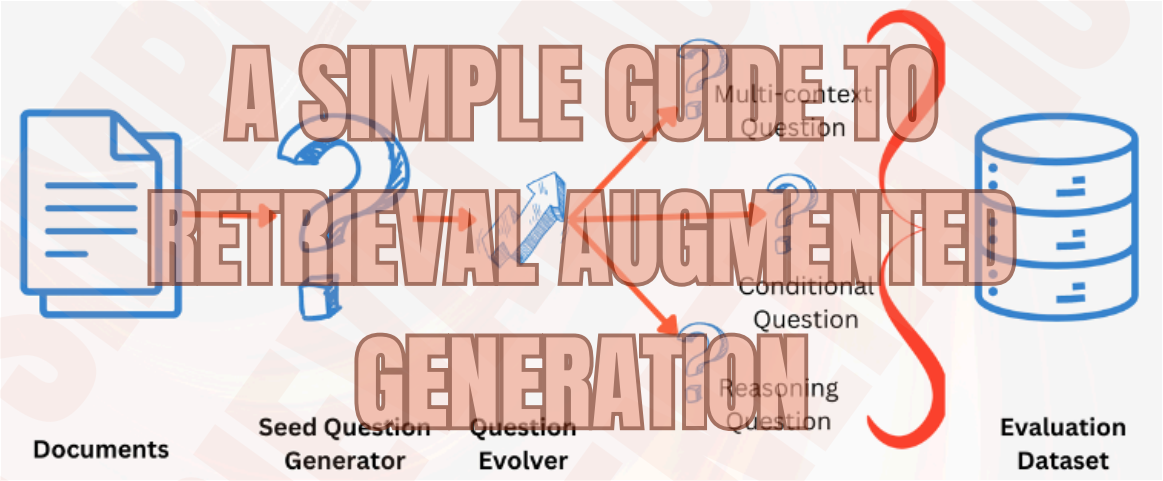


Figure 13: Synthetic Test Dataset Generation

ARES: Automated RAG evaluation system, or ARES, is a framework developed by researchers at Stanford University and Databricks. Like RAGAs, ARES uses an LLM as a judge approach for evaluations.

Benchmarks: Standardised datasets and their evaluation metrics used to measure the performance of RAG systems. Benchmarks provide a common ground for comparing different RAG approaches. Benchmarks ensure consistency across the evaluations by considering fixed tasks and evaluation criteria.

BEIR or Benchmarking Information Retrieval: A comprehensive, heterogeneous benchmark that is based on 9 IR tasks and 19 Question-Answer datasets.

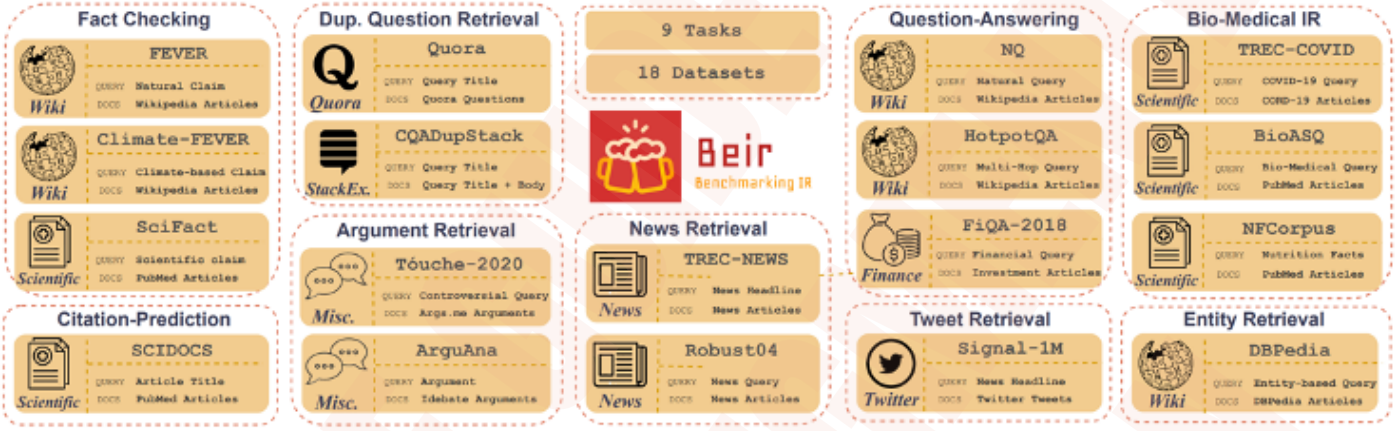


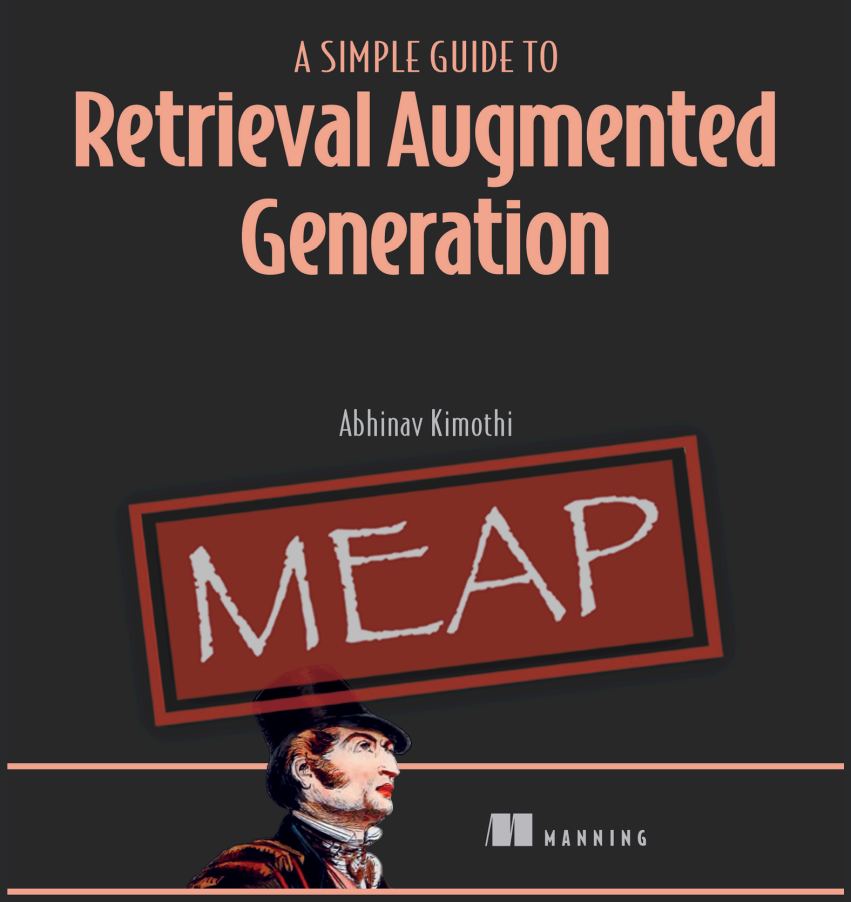
Figure 14: BEIR – 9 tasks and 18 (of 19) datasets (Source: BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models)

Retrieval Augmented Generation Benchmark (RGB): Introduced in a Dec 2023, it comprises 600 base questions and 400 additional questions, evenly split between English and Chinese. It is a benchmark that focusses on four key abilities of a RAG system – Noise Robustness, Negative Rejection, Information Integration and Counterfactual Robustness

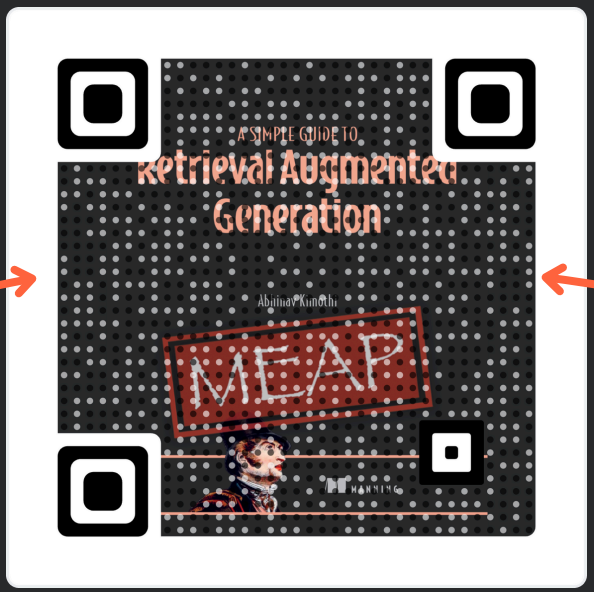
Multihop RAG: Curated by researchers at HKUST, Multihop RAG contains 2556 queries, with evidence for each query distributed across 2 to 4 documents. The queries also involve document metadata, reflecting complex scenarios commonly found in real-world RAG applications.

Comprehensive RAG: Curated by Meta and HKUST, CRAG, is a factual question answering benchmark of 4,409 question-answer pairs and mock APIs to simulate web and Knowledge Graph (KG) search. It contains 8 types of queries across 5 domains

THIS TAXONOMY IS BASED ON



IF YOU LIKE THIS TAXONOMY, CONSIDER PURCHASING THE BOOK



SCAN CODE OR CLICK HERE TO GET AN EARLY ACCESS COPY

Naive RAG: A basic linear approach to RAG with sequential indexing, retrieval, augmentation and generation process.

Retrieve-Read: A retriever that retriever reads information and the LLM that is reads this information to generate the results

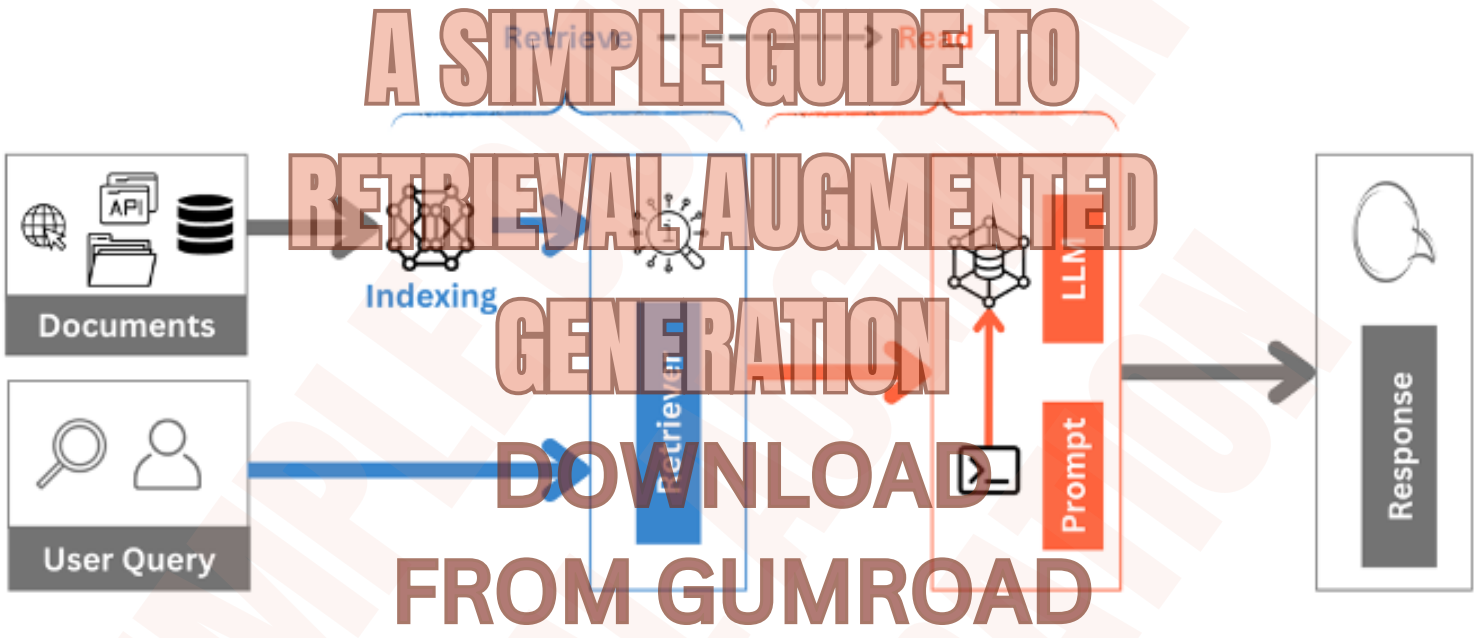


Figure 15: Naïve RAG is a sequential "Retrieve-Read" process.

RAG Failure Points: A RAG system can misfire if the the retriever fails to retrieve the entire context or retrieves irrelevant context, the LLM despite being provided the context, does not consider it and, instead of answering the query picks irrelevant information from the context.

Disjointed Context: When information is retrieved from different source documents, the transition between two chunks becomes abrupt.

Over-reliance on Context: It is noticed, sometimes, that the LLM becomes over-reliant on the retrieved context and forgets to draw from its own parametric memory.

Advanced RAG: Pipeline with interventions at pre-retrieval, retrieval and post-retrieval stages to overcome the limitations of Naive RAG.

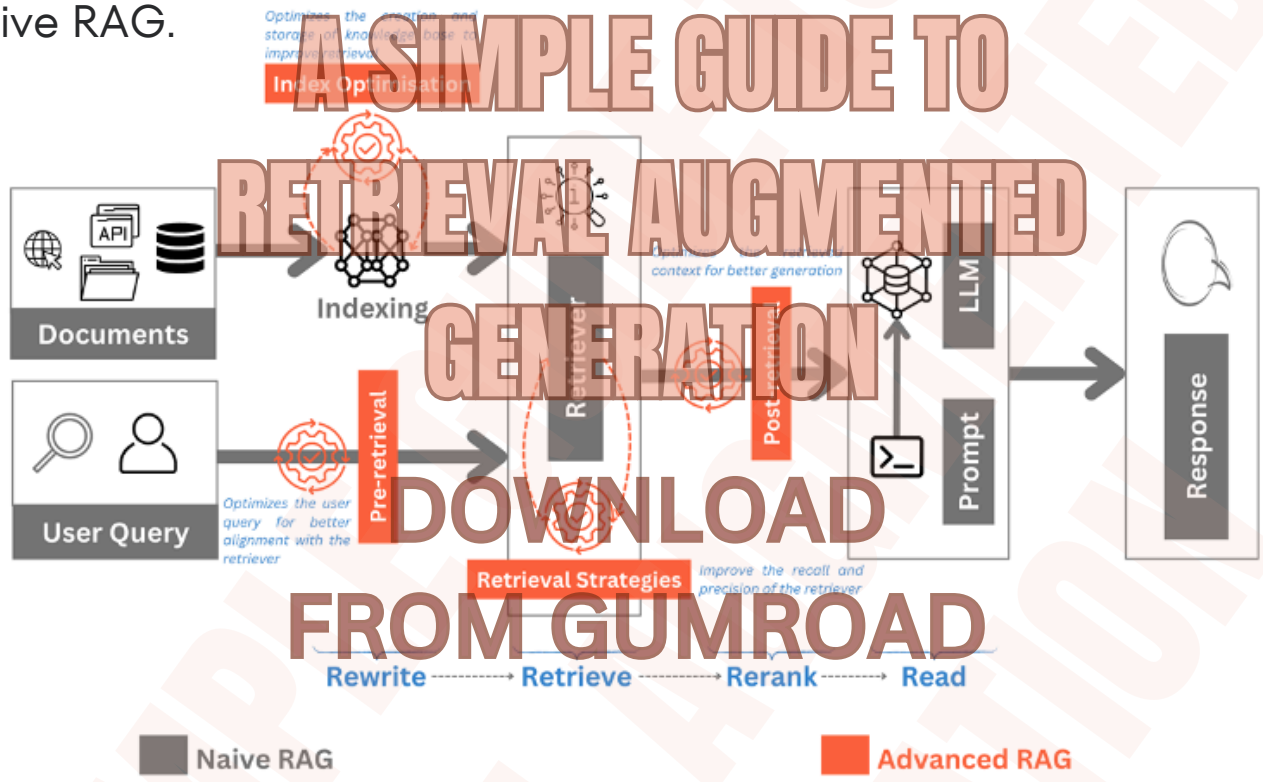


Figure 16: Advanced RAG is a Rewrite-Retrieve-Rerank-Read process as compared to a Retrieve-Read Naïve RAG process

Rewrite-Retrieve-Rerank-Read: Improvement upon Retrieve-Read framework by adding the rewriting and reranking component

Index Optimisation: Employed in the indexing pipeline, the objective of index optimisation is to set up the knowledge base for better retrieval.

Query Optimisation: The objective of query optimisation is to align the input user query in a manner that makes it better suited for the retrieval tasks

Query Expansion: The original user query is enriched with the aim of retrieving more relevant information. This helps in increasing the recall of the system and overcomes the challenge of incomplete or very brief user queries.

Multi-query expansion: In this approach, multiple variations of the original query are generated using an LLM and each variant query is used to search and retrieve chunks from the knowledge base.

Sub-query expansion: In this approach instead of generating variations of the original query, a complex query is broken down into simpler sub-queries.

Step back expansion: The term comes from the step-back prompting approach where the original query is abstracted to a higher-level conceptual query.

Query Transformation: In query transformation, instead of the original user query retrieval happens on a transformed query which is more suitable for the retriever

Query Rewriting: Queries are rewritten from the input. The input in quite a few real-world applications may not be a direct query or a query suited for retrieval. Based on the input a language model can be trained to transform the input into a query which can be used for retrieval.

Hypothetical document embedding: HyDE is a technique where the language model first generates a hypothetical answer to the user's query without accessing the knowledge base. This generated answer is then used to perform a similarity search against the document embeddings in the knowledge base, effectively retrieving documents that are similar to the hypothetical answer rather than the query itself.

Query Routing: Optimising the user query by routing it to the appropriate workflow based on criteria like intent, domain, language, complexity, source of information etc

Hybrid Retrieval: Hybrid retrieval strategy is an essential component of production-grade RAG systems. It involves combining retrieval methods for improved retrieval accuracy. This can mean simply using a keyword-based search along with semantic similarity. It can also mean combining all sparse embedding, dense embedding vector and knowledge graph-based search.

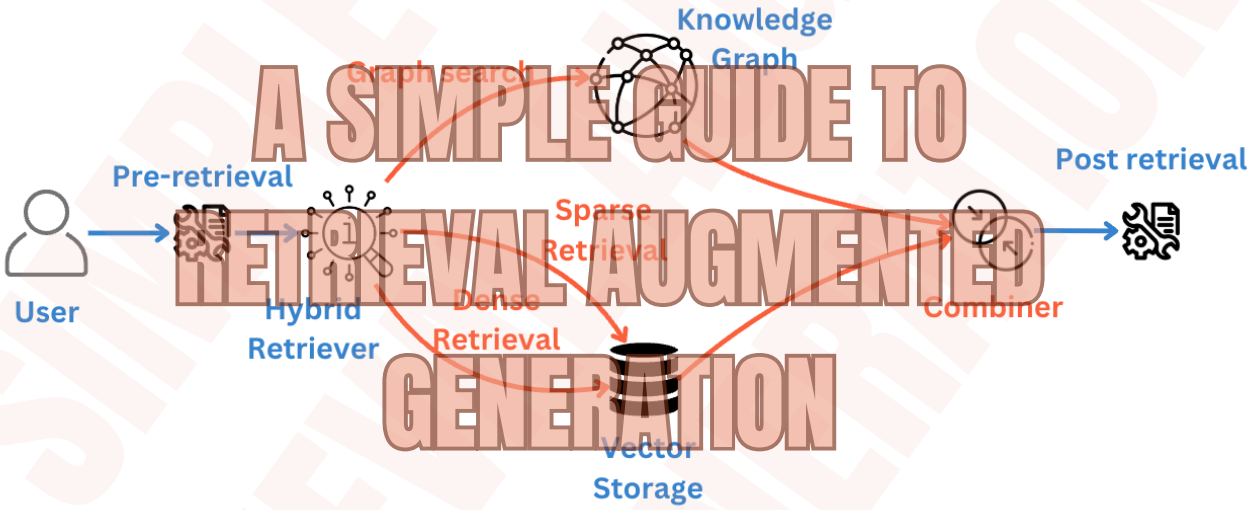


Figure 17: Hybrid retriever employs multiple querying techniques and combines the results

Iterative Retrieval: In this approach the retrieval happens 'n' number of times and the generated response is used to further retrieve documents each time.

Recursive Retrieval: This approach improves upon the iterative approach by transforming the retrieval query after each generation.

Adaptive Retrieval: This is a more intelligent approach where an LLM determines the most appropriate moment and the most appropriate content for retrieval.

Contextual Compression: Reduce the length of the retrieved information by extracting only the parts that are relevant and important to the query. This also has a positive effect on the cost and efficiency of the system.

Reranking: Retrieved information from different sources and techniques can further be ranked to determine the most relevant documents. Reranking, like hybrid retrieval, is commonly becoming a necessity in production RAG systems. To this end, commonly available rerankers like multi-vector, Learning to Rank (LTR), BERT based and even hybrid rerankers that can be employed are gaining prominence.

Modular RAG: Modular RAG breaks down the traditional monolithic RAG structure into interchangeable components. This allows for tailoring of the system to specific use cases. Modular approach brings modularity to RAG components like retrievers, indexing, and generation, while also adding more modules like search, memory, and fusion.

Search Module: Aimed at performing search on different data sources, it is customised to different data sources.

RAG-Fusion: Improves traditional search systems by overcoming their limitations through a multi-query approach.

Memory Module: The Memory Module leverages the inherent 'memory' of the LLM, meaning the knowledge encoded within its parameters from pre-training.

Routing: 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.

Task Adapter: This module makes RAG adaptable to various downstream tasks allowing the development of task-specific end-to-end retrievers with minimal examples, demonstrating flexibility in handling different tasks. The Task Adapter Module allows the RAG system to be fine-tuned for specific tasks like summarisation, translation, or sentiment analysis.

Critical Layers: Fundamental components for the operation of a RAG system. A RAG system is likely to fail if any of these layers are missing or incomplete

Data Layer: The data layer serves the critical role of creating and storing the knowledge base for RAG. It is responsible for collecting data from source systems, transforming it into a usable format and storing it for efficient retrieval.

Model Layer: Predictive models enable generative AI applications. Some models are provided by third parties and some need to be custom trained or fine-tuned. Generating quick and cost-effective model responses is also an important aspect of leveraging predictive models. This layer holds the model library, training & fine-tuning and the inference optimisation components.

Fully managed deployment: Deployment provided by service providers where all infrastructure for model deployment, serving, and scaling is managed and optimised by these providers

Self-hosted deployment: In this scenario, models are deployed in private clouds or on-premises, and the infrastructure is managed by the application developer. Tools like Kubernetes and Docker are widely used for containerisation and orchestration of models, while Nvidia Triton Inference Server can optimise inference on Nvidia hardware.

Local/edge deployment: Running optimised versions of models on local hardware or edge devices, ensuring data privacy, reduced latency, and offline functionality.

Application Orchestration Layer : Layer responsible for managing the interactions amongst the other layers in the system. It is a central coordinator that enables communication between data, retrieval systems, generation models and other services.

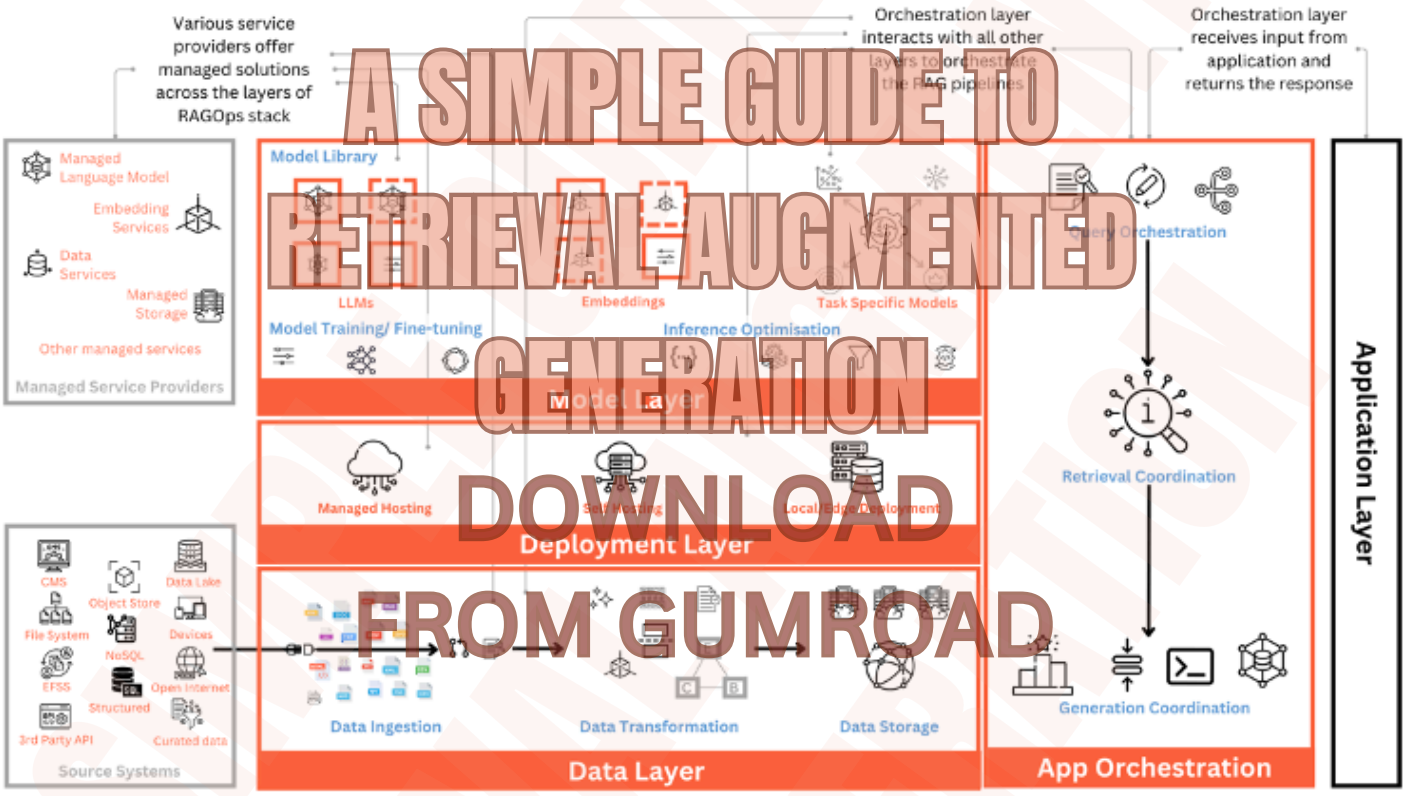


Figure 18: Core RAGOps stack where Data, Model, Model Deployment and App Orchestration layers interact with source systems and managed service providers and co-ordinate with the application layer to interface with the user.

Essential Layers: Layers focussing on performance, reliability and safety of the system. These essential components bring the system to a standard that provides value to the user.

Prompt Layer: Manages the augmentation and other LLM prompts.

Evaluation Layer: Manages regular evaluation of retrieval accuracy, context relevance, faithfulness and answer relevance of the system is necessary to ensure the quality of responses.

Monitoring Layer: Continuous monitoring ensures the long-term health of the RAG system. Understanding system behaviour and identifying points of failure, assessing the relevance & adequacy of information, and tracking regular system metrics tracking like resource utilisation, latency and error rates form the part of the monitoring layer.

LLM Security & Privacy Layer: RAG systems rely on large knowledge bases stored in vector databases, which can contain sensitive information. They need to follow all data privacy regulations, data protection strategies like anonymisation, encryption, differential privacy, query validation & sanitisation, and output filtering to assist in protection against attacks. Implementing guardrails, access controls, monitoring and auditing are also components of the security and privacy layer.

Caching Layer: Caching is becoming a very important component of any LLM based application. This is because the high costs and inherent latency of generative AI models. With the addition of retrieval layer, the costs and latency increase further in RAG systems.

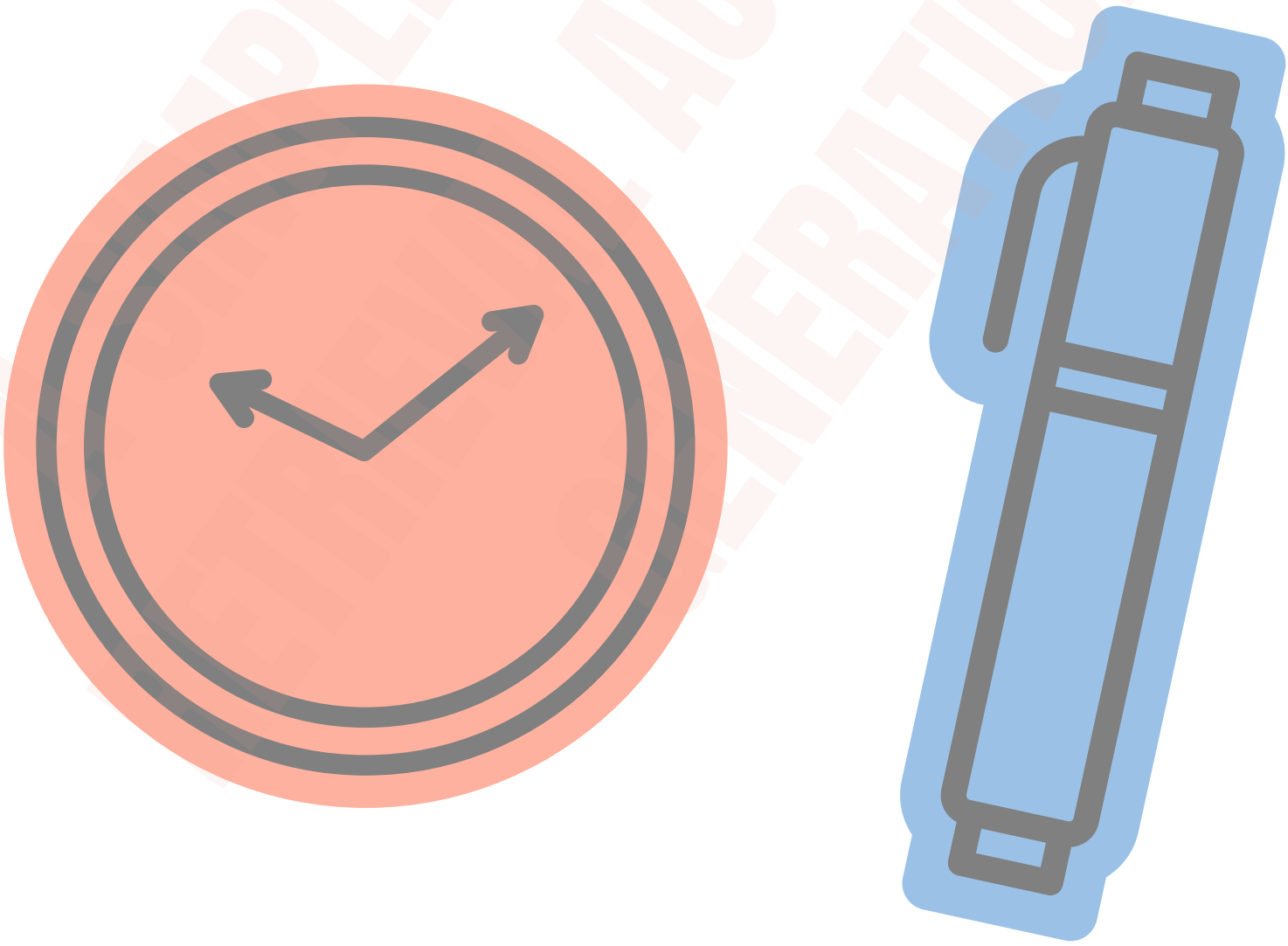
Enhancement Layer: Layers improving the efficiency, scalability and usability of the system. These components are used to make the RAG system better and are decided based on the end requirements.

Human-in-the-loop Layer: Provides critical oversight where human judgment is necessary, especially for use-cases requiring higher accuracy or ethical considerations.

Cost Optimisation Layer: Helps manage resources efficiently, which is particularly important for large-scale systems.

Explainability and Interpretability Layer: Provides transparency for system decisions, especially important for domains requiring accountability

Collaboration and Experimentation Layer: Useful for teams working on development and experimentation but non-critical for system operation.



Knowledge Graph powered RAG: Using knowledge graph structures not only increases the contextual understanding but also equips the system with enhanced reasoning capabilities and improved explainability.

Knowledge Graphs: Knowledge graphs organise data in a structured manner as entities and relationships.

GraphRAG: An open-source framework developed by Microsoft that facilitates automatic creation of knowledge graphs from source documents and then uses the knowledge graph for retrieval.

Graph Communities: Partitioning entities & relationships into groups.

Community Summaries: LLM generated summaries for communities, providing insights into topical structure and semantics

Local Search: Identifying a set of entities from the knowledge graph that are semantically-related to the user input

Global Search: Similarity based search on community summaries.

Ontology: A formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts.

Multimodal RAG: Using other modalities like images, audio, video etc. in addition to text in both retrieval and generation.

Modality: A specific type of input data, such as text, image, video, or audio. Multimodal systems can handle multiple modalities simultaneously.

Multimodal Embeddings: A unified vector representation that encodes multiple data types (e.g., text and image embeddings combined) to enable retrieval across different modalities.

CLIP (Contrastive Language-Image Pre-training): A model developed by OpenAI that learns visual concepts from natural language supervision, often used for cross-modal retrieval and generation

Contrastive Learning: A learning method used to align data across different modalities by bringing semantically similar data points closer in the shared embedding space.

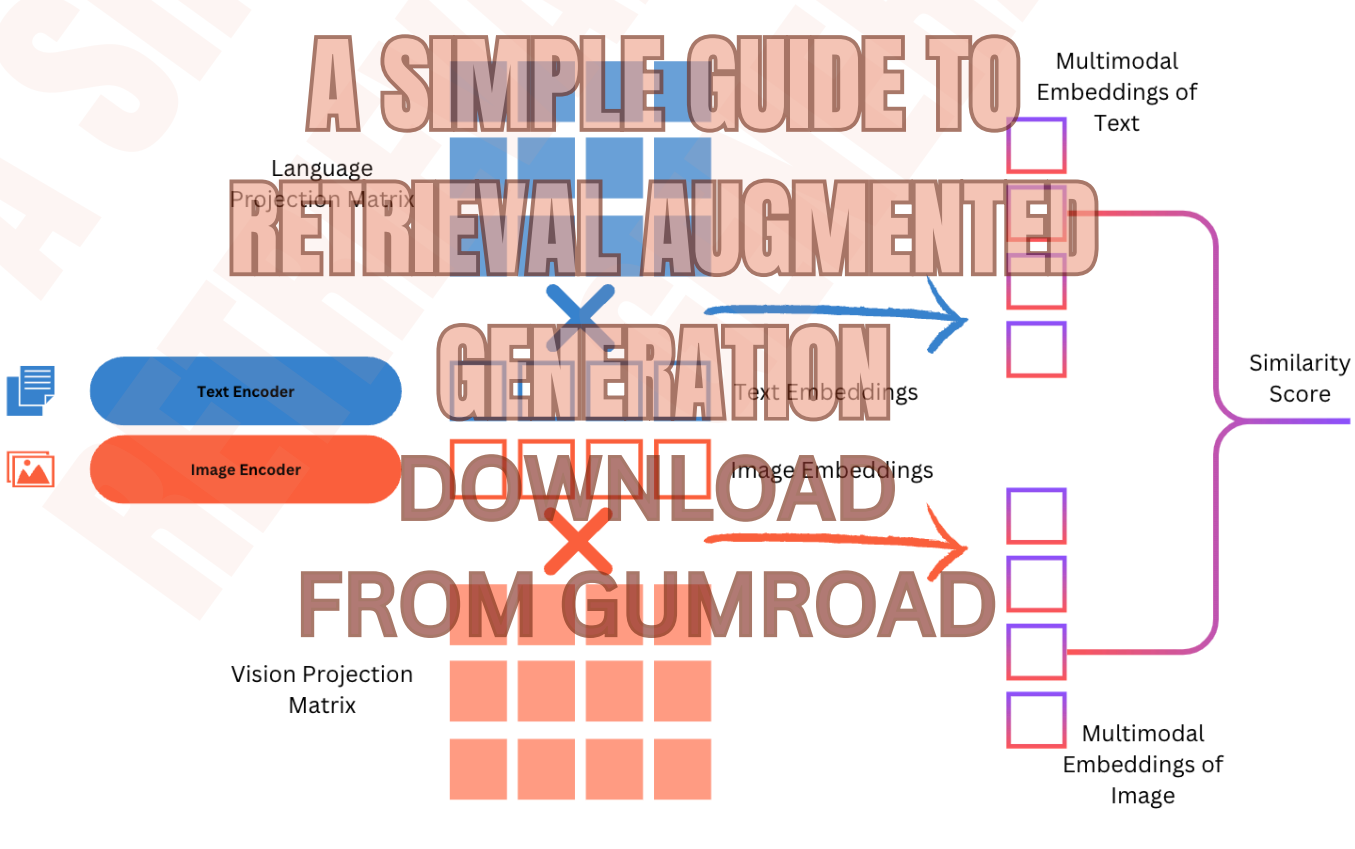


Figure 19: Mapping data of different modalities into a shared embedding space

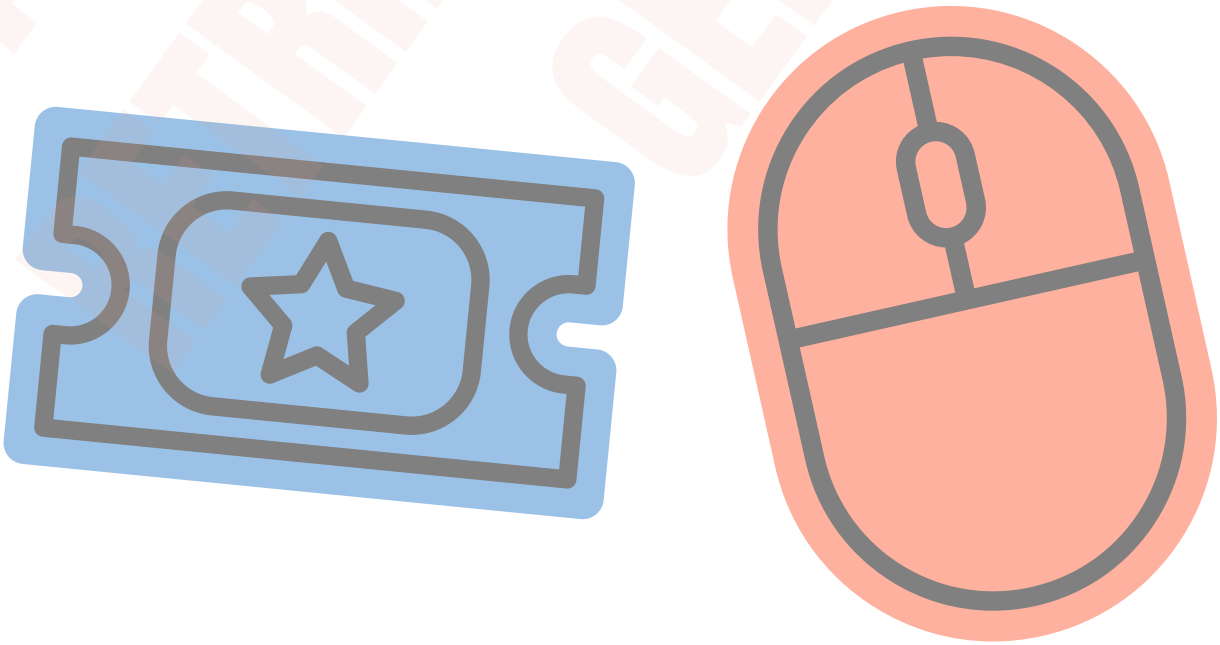
Agentic RAG: Leverages LLM based agents for adapting RAG workflow to query types and the type of documents in the knowledge base.

Adaptive Frameworks: Dynamic systems that adjust retrieval and generation strategies based on the evolving context and data, ensuring relevant responses.

Routing Agents: Agents responsible for directing user queries to the most appropriate sources or sub-systems for efficient processing.

Query Planning Agents: Agents that break down complex queries into sub-queries and manage their execution across different retrieval pipelines.

Multiple Vectors per Document: A technique where multiple vector representations are generated for each document to capture different aspects of its content.



Technology Providers

Model Access, Training & FineTuning

- OpenAI
- HuggingFace
- Google Vertex AI
- Anthropic
- AWS Bedrock
- AWS Sagemaker
- Cohere
- Azure Machine Learning
- IBM Watson AI
- Mistral AI
- Salesforce Einstein
- Databricks Dolly
- NVIDIA NeMo
- EleutherAI

Data Loading

- Snorkel AI
- LlamaIndex
- LangChain
- Scale AI
- Labelbox
- Superb AI
- Explorium
- Roboflow
- Datature
- V7 Labs
- Clarifai

Vector DB and Indexing

- Pinecone
- Milvus
- Chroma
- Weaviate
- Deep Lake
- Qdrant
- Elasticsearch
- Vespa
- Redis (Vector Search Support)
- Vald
- Zilliz
- Marqo
- PGVector (PostgreSQL extension)
- MongoDB (with vector capabilities)
- SingleStore

Application Framework

- LangChain
- LlamaIndex
- Haystack
- CrewAI (Agentic Orchestration)
- AutoGen (Agentic Orchestration)
- LangGraph (Agentic Orchestration)
- Rasa (Conversational AI)
- Flyte
- Prefect
- Airflow
- Metaflow

Technology Providers

7

Prompt Engineering

- W&B (Weights & Biases)
- PromptLayer
- TruLens
- TruEra
- PromptHero
- TextSynth

Deployment Frameworks

- Vllm
- TensorRT-LLM
- ONNX Runtime
- KubeFlow
- MLflow
- Ray Serve
- Triton Inference Server
- Seldon Deploy

Deployment & Inferencing

- AWS
- GCP
- OpenAI API
- Azure
- IBM Cloud
- Oracle Cloud Infrastructure
- Heroku
- Kubernetes
- DigitalOcean
- Vercel

Monitoring

- HoneyHive
- TruEra
- Fiddler AI
- Arize AI
- Aporia
- WhyLabs
- Evidently AI
- Superwise
- Monte Carlo
- Datadog

Proprietary LLMs/VLMs

- GPT series by OpenAI
- Gemini series by Google
- Claude series by Anthropic
- Command series by Cohere
- Jurassic by AI21 Labs
- PalM by Google
- LaMDA by Google

Open Source LLMs

- Llama series by Meta
- Mixtral by Mistral
- Falcon by TII
- Vicuna by LMSYS
- GPT-NeoX by EleutherAI
- Pythia by EleutherAI
- Dolly 2.0 by Databricks
- Phi by Microsoft

Technology Providers

Small Language Models

- Phi by Microsoft
- GPT-Neo by EleutherAI
- DistilBERT by HuggingFace
- TinyBERT
- ALBERT (A Lite BERT) by Google
- MiniLM by Microsoft
- DistilGPT2 by HuggingFace
- Reformer by Google
- T5-Base by Google

Managed RAG solutions

- OpenAI File Search
- Amazon Bedrock Knowledge Bases
- Azure AI File Search
- Claude Projects
- Vectorize.io

Knowledge Graph and Ontology

- Neo4j
- Stardog
- TerminusDB
- TigerGraph

Security and Privacy

- Hazy
- Duality
- BigID

Synthetic Data

- Mostly AI
- Tonic.ai
- Synthesis AI

Others

- Cohere reranker
- Unstructured.io

Other RAG Patterns

Corrective RAG: In this approach, real-time information is retrieved to check for the factual accuracy of the LLM generated answer. Particularly useful in fact-checking, medical & legal domains.

Contrastive RAG: Integrates contrastive learning techniques to enhance the retrieval process by distinguishing between relevant and irrelevant documents. https://arxiv.org/abs/2406.06577

Selective RAG: Optimises the retrieval phase by determining when it is beneficial to retrieve external information. This method aims to improve the overall performance of language models, particularly in contexts where retrieval may not add value.

RAG with Active Learning: User feedback on generated content is used to fine-tune or adapt the retrieval process over time. Useful in continuous improvement systems, like recommendation engines or educational tutoring systems, where the goal is to enhance performance with each interaction.

Personalised RAG: User preferences, behaviour, and historical interactions are used to personalise the retrieval process. It's used in personalisation-heavy domains like recommendation engines or customer service, where the system tailors responses to individual users.

Self-RAG: An adaptive retrieval mechanism that selectively decides when to retrieve knowledge based on the query's context.

RAFT: Retrieval-Augmented Fine-Tuning combine retrieval mechanisms with traditional fine-tuning techniques to help models access and utilise external knowledge dynamically during their fine-tuning process.

RAPTOR: Recursive Abstractive Processing for Tree-Organised Retrieval focusses on creating a recursive, tree-like structure from documents to improve context-aware information retrieval. It is beneficial for question-answering tasks, especially when dealing with extensive documents or information that requires multi-step reasoning.

Application Areas

Search Engine: Conventional search results are shown as a list of page links ordered by relevance. More recently, Google Search, Perplexity, You have used RAG to present a coherent piece of text, in natural language, with source citation. As a matter of fact, search engine companies are now building LLM first search engines where RAG is the cornerstone of the algorithm.

Personalised Marketing Content Generation: The widest use of LLMs has probably been in content generation. Using RAG, the content can be personalised to readers, incorporate real-time trends and be contextually appropriate. Yarnit, Jasper, Simplified are some of the platforms that assist in marketing content generation like blogs, emails, social media content, digital advertisements etc.

Personalised Learning Plans: In the field or education, and learning & development, RAG is used extensively to create personalised learning paths based on past trends and for automated evaluation and feedback.

Real-time Event Commentary: Imagine an event like a sport or a news event. A retriever can connect to real-time updates/data via APIs and pass this information to the LLM to create a virtual commentator. These can further be augmented with Text-To-Speech models. IBM leveraged technology for commentary during the 2023 US Open Tennis tournament.

Conversational agents: LLMs can be customised to product/service manuals, domain knowledge, guidelines, etc. using RAG and serve as support agents resolving user complaints and issues. These agents can also route users to more specialised agents depending on the nature of the query. Almost all LLM based chatbots on websites or as internal tools use RAG. These are being used in industries like Travel & Hospitality, Fintech and e-commerce.

Document Question Answering Systems: With access to proprietary documents, a RAG enabled system becomes an intelligent AI system that can answer all questions about the organisation.

Virtual Assistants: Virtual personal assistants like Siri, Alexa and others are in plans to use LLMs to enhance the user's experience. Coupled with more context on user behaviour using RAG, these assistants are set to become more personalised.

Applied RAG Challenges

Relevance Mismatch: Difficulty retrieving the most relevant documents or passages from a large dataset due to suboptimal ranking or search mechanisms.

Over-Retrieval: Retrieving too many documents, leading to unnecessary noise and irrelevant content in the final generation.

Sparse vs Dense Retrieval Trade-off: Balancing between sparse retrieval (TF-IDF, BM25) and dense retrieval (using embeddings) to maximise relevance without losing performance.

Document Question Answering Systems: With access to proprietary documents, a RAG enabled system becomes an intelligent AI system that can answer all questions about the organisation.

Latency: Retrieval from large or distributed knowledge bases can introduce significant delays, affecting real-time applications.

Cost of Storage: Maintaining and managing massive vector databases for dense retrieval can be expensive and resource-intensive.

Narrow Retrieval Focus: The challenge of retrieving diverse perspectives or sources to ensure that multiple viewpoints or alternative pieces of evidence are surfaced.

Bias in Retrieval: Biases in retrieval results based on the structure of the indexed data or retrieval algorithms.

Context Loss in Long Queries: Loss of context when handling long, multi-turn queries, leading to disjointed or incomplete answers.

Incoherent Summarisation: Generating inconsistent or disjointed summaries from multiple retrieved documents, leading to poor user experience.

Over-Generation: Generating overly verbose or redundant responses based on retrieved data that fails to condense the key points effectively.

Inconsistent Modal Alignment: Challenges integrating multimodal data (e.g., text, images, videos) where retrieved content across different modalities may not be aligned properly, affecting the quality of the generated response.

Data Silos: When knowledge is fragmented across multiple sources or platforms, retrieval becomes challenging due to inconsistent indexing or lack of interoperability between knowledge bases.

Processing Large-Scale Data: Difficulty in maintaining high throughput and low latency as the amount of indexed data grows.

Multi-Agent Coordination: When multiple agents are used for task orchestration (as in agentic RAG), coordination among agents can become complex and resource-heavy, impacting system efficiency.

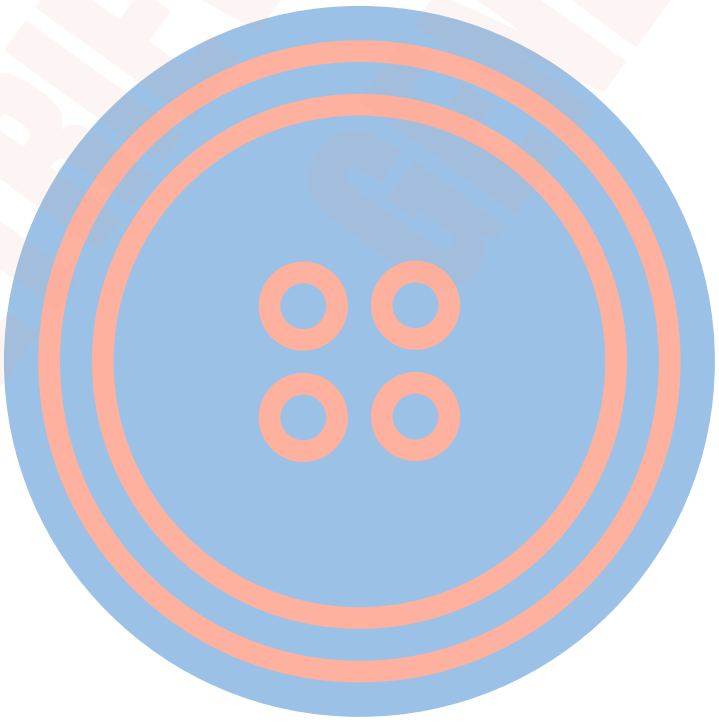
Inefficient Query Routing: Routing queries to the wrong sources or retrieval pipelines can reduce efficiency.

Data Poisoning Attacks: External retrieval sources could be manipulated to feed biased or poisoned data into the generation pipeline, leading to compromised outputs.

Adversarial Attacks: Security vulnerabilities where attackers may influence retrieval or generation results by exploiting weaknesses in retrieval pipelines.

Knowledge Base Updating: Maintaining an up-to-date knowledge base while keeping the retrieval process fast and accurate can be difficult, especially in dynamic fields like news or finance.

Memory Retention: Ensuring that the system can store and retrieve long-term memory across interactions, allowing for a more personalised and context-aware response.

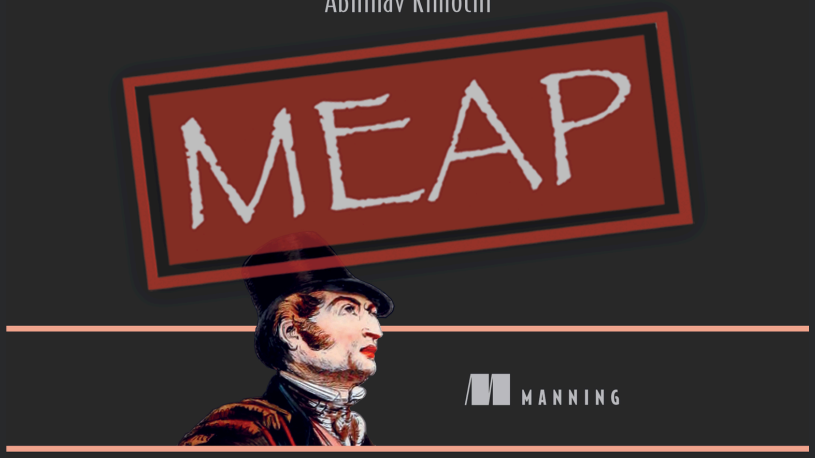


THIS TAXONOMY IS BASED ON

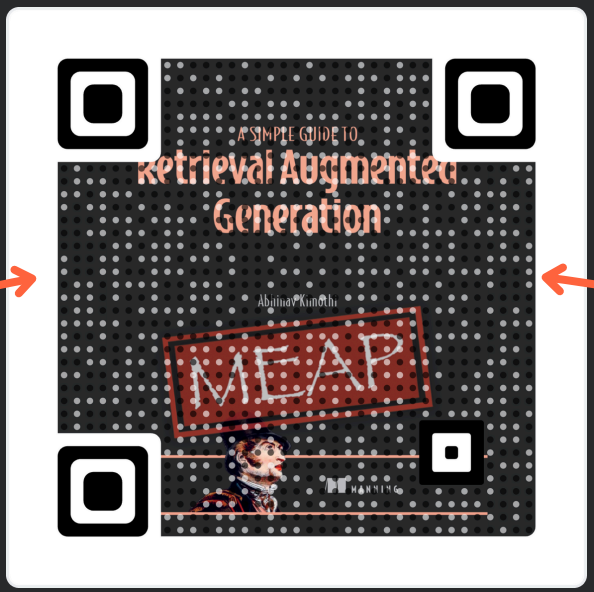
A SIMPLE GUIDE TO

Retrieval Augmented Generation

Abhinav Kimothi



IF YOU LIKED THIS TAXONOMY, CONSIDER PURCHASING THE BOOK



SCAN CODE OR CLICK HERE TO GET AN EARLY ACCESS COPY

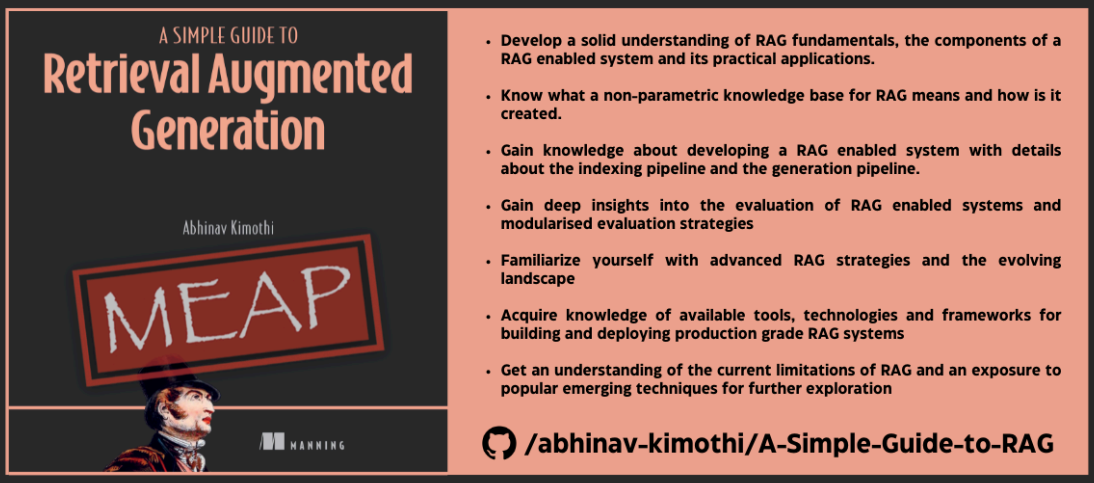
INTERESTED IN CODING RAG PIPELINES?

THE SOURCE CODE OF

A SIMPLE GUIDE TO

RETRIEVAL AUGMENTED GENERATION

ARE NOW AVALIABLE FOR FREE PUBLIC ACCESS

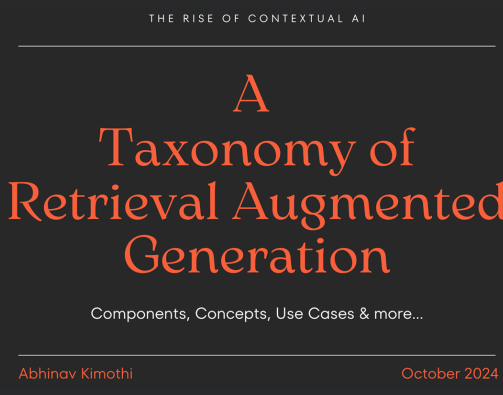


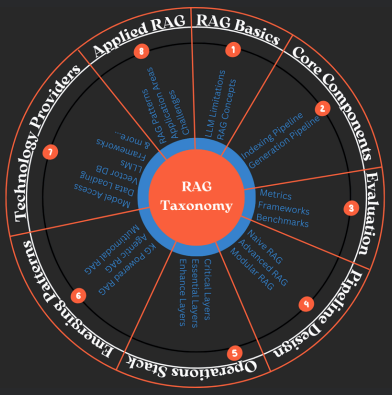
MANNING

GitHub

SCAN CODE OR CLICK HERE TO VIEW SOURCE CODE

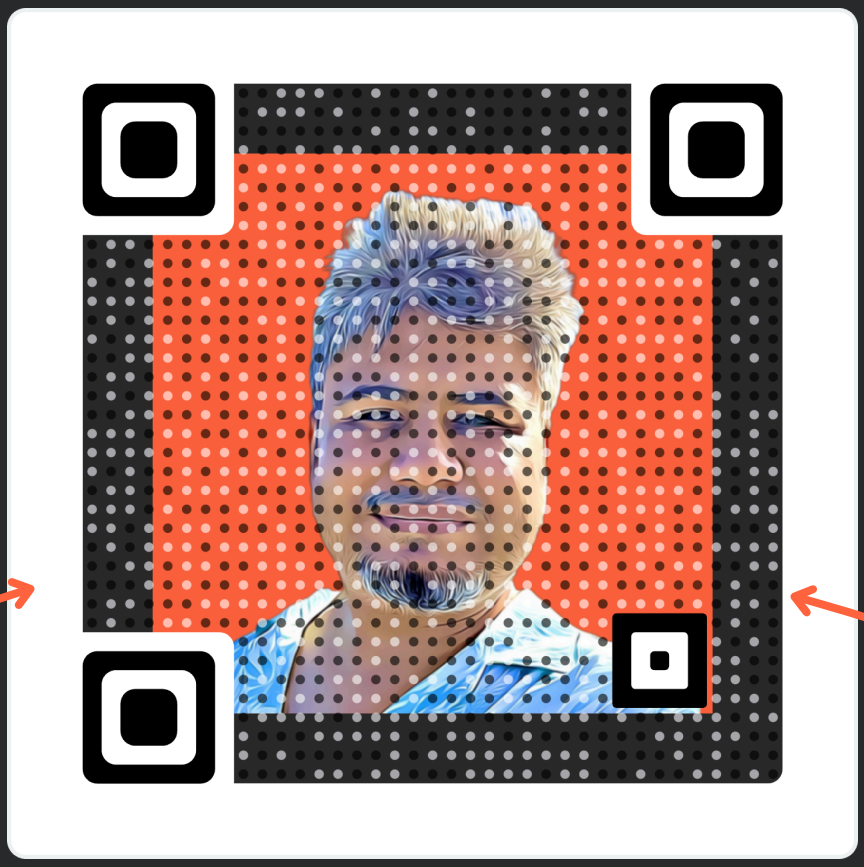
A DOWNLOADABLE PDF VERSION OF THIS EBOOK IS ALSO AVAILABLE ON GUMROAD





SCAN CODE OR CLICK HERE TO DOWNLOAD

Hi! My name is Abhinav, and I talk about ML, RAG, LLMs & AI Agents. If our interests align, I'd love to stay connected



SCAN CODE OR CLICK HERE TO CONNECT

linktr.ee/abhinavkimothi

/in/Abhinav-Kimothi

@akaiworks

@abhinav_kimothi

@abhinavkimothi