Part 1 Foundation

# 1 Large Language Models and the Need for Retrieval Augmented Generation

### This chapter covers

* What is Retrieval Augmented Generation?
* What are Large Language Models and How are they used?
* The challenges with Large Language Models and the need for RAG
* Popular use cases of RAG

In a short time, Large Language Models have found a wide applicability in modern language processing tasks and even paved the way for autonomous AI agents. There are high chances that you’ve heard about, if not personally used, ChatGPT. ChatGPT is powered by a generative AI technique called Large Language Models. Retrieval Augmented Generation, or RAG, plays a pivotal role in the application of Large Language Models by enhancing their memory and recall.

This book aims to demystify the idea and application of Retrieval Augmented Generation. Over the course of this book, you will be presented with the definition, the design, implementation, evaluation, and the evolution of this technique.

To kick things off, in this chapter, we will introduce the concepts behind Retrieval Augmented Generation and investigate its pressing need with the help of some examples. We will also take a brief look at Large Language Models and how one can interact with them. We will, further, discuss the challenges inherent to Large Language Models, how Retrieval Augmented Generation overcomes these challenges and, at the end, list down a few use cases that have been enabled by this technique.

By the end of this chapter, you will have gained a foundational knowledge to be ready for a deeper exploration of the components of a RAG-enabled system.

By the end of this chapter, you should –

* Have a strong hold on the definition of Retrieval Augmented Generation.
* Develop a basic level of familiarity with Large Language Models.
* Be able to appreciate the limitation of LLMs and the need for RAG.
* Be equipped to dive into the components of a RAG enabled system.

Retrieval Augmented Generation

Generative AI models struggle when you ask them about facts not covered in their training data. Retrieval Augmented Generation—or RAG—enhances an LLM’s available data by adding context from an external knowledge base, so it can answer accurately about proprietary content, recent information, and even live conversations.

In a short time, Large Language Models have found a wide applicability in modern language processing tasks and even paved the way for autonomous AI agents. There are high chances that you’ve heard about, if not personally used, ChatGPT. ChatGPT is powered by a generative AI technique called Large Language Models. Retrieval Augmented Generation, or RAG, plays a pivotal role in the application of Large Language Models by enhancing their memory and recall.

This book aims to demystify the idea and application of Retrieval Augmented Generation. Over the course of this book, you will be presented with the definition, the design, implementation, evaluation, and the evolution of this technique.

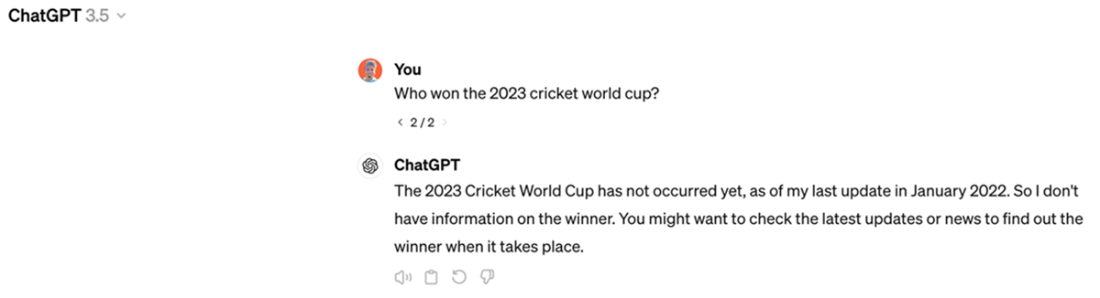
To kick things off, in this chapter, we will introduce the concepts behind Retrieval Augmented Generation and investigate its pressing need with the help of some examples. We will also take a brief look at Large Language Models and how one can interact with them. We will, further, discuss the challenges inherent to Large Language Models, how Retrieval Augmented Generation overcomes these challenges and, at the end, list down a few use cases that have been enabled by this technique.

By the end of this chapter, you will have gained a foundational knowledge to be ready for a deeper exploration of the components of a RAG-enabled system.

What is RAG?

Retrieval Augmented Generation, or RAG, has emerged to be one of the most popular techniques in the applied generative AI world. Large Language Models, or LLMs, is a generative AI technology that has recently gained tremendous popularity. The most common example of the application of an LLM is ChatGPT by OpenAI. LLMs, like the one powering ChatGPT, have been shown to store knowledge in them. You can ask them questions and they tend to respond with answers that seem correct. However, despite their unprecedented ability to generate text, their responses are not always correct. Upon more careful observation, you may notice that LLM responses are plagued with sub-optimal information and inherent memory limitations. RAG addresses these limitations of LLMs by providing them with information external to these models. Thereby, resulting in LLM responses that are more reliable and trustworthy.

To understand the basic concept of RAG, we will use a simple example. Those familiar with the wonderful sport of Cricket will recall that the Men’s ODI Cricket World Cup tournament was held in 2023. The Australian cricket team emerged as the winner. Now, imagine you are interacting with ChatGPT, and you ask it a question, say, “Who won the 2023 Cricket World Cup?”. You are, in truth, interacting with GPT-3.5 or GPT-4, LLMs developed and maintained by OpenAI that power ChatGPT. In the first few sections of this chapter, we will use ChatGPT and LLMs interchangeably for simplicity. So, you ask the question and, most likely, you will observe a response like the one illustrated in figure 1.1 below.



##### Figure 1.1 ChatGPT response to the question, “Who won the 2023 cricket world cup?” (Variation 1), Source: Screenshot by author of his account on [https://chat.openai.com](https://chat.openai.com/)

ChatGPT does not have any memory of the 2023 Cricket World Cup and it tells you to check the information from other sources. This is not ideal but, at least, ChatGPT is honest in its response. The same question asked again might also provide a factually inaccurate result. Look at the following illustration in figure 1.2. ChatGPT, falsely, responds that India was the winner of the tournament.



##### Figure 1.2 ChatGPT response to the question, “Who won the 2023 cricket world cup?” (Variation 2), Source: Screenshot by author of his account on [https://chat.openai.com](https://chat.openai.com/)

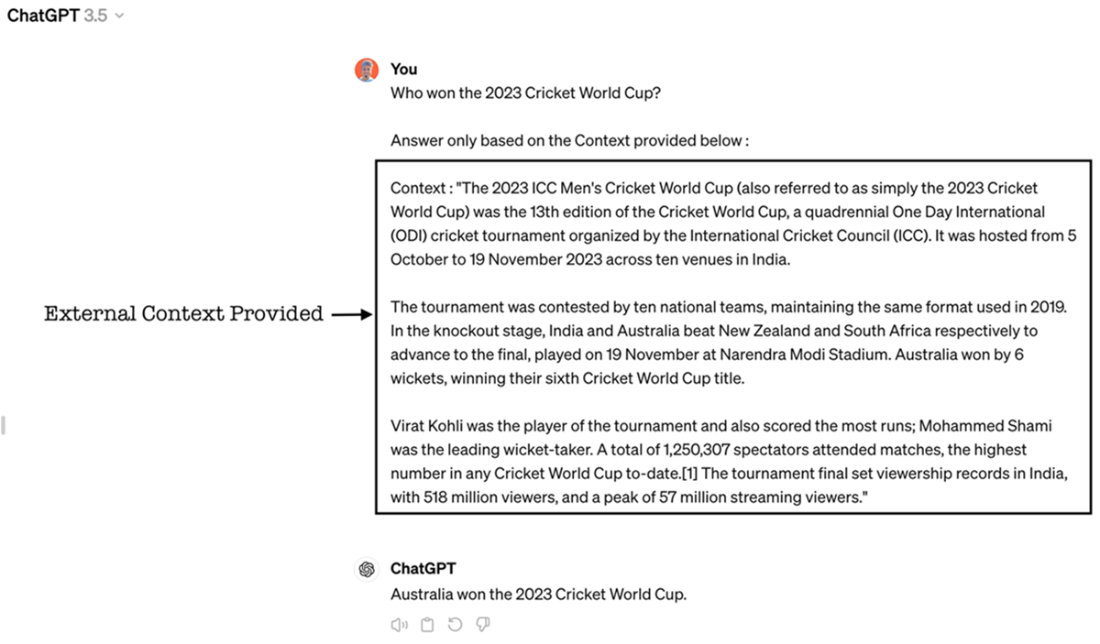
This is problematic. Despite not having any memory of the 2023 Cricket World Cup, ChatGPT still generates the answer, in a seemingly confident tone, but does that inaccurately. This is what is called a “hallucination” and this has become a major point of criticism for LLMs.

What can be done to improve the response? The world, of course, has this knowledge about the 2023 Cricket World Cup. A simple Google Search will tell you about the winner of the 2023 Cricket World Cup, if you don’t already know it. The Wikipedia article (figure 1.3) on the 2023 Cricket World Cup accurately provides this information in the opening section itself. If only, there was a way to tell the LLM about the 2023 Cricket World Cup.



Figure 1.3 Wikipedia Article on 2023 Cricket World Cup, Source : <https://en.wikipedia.org/wiki/2023_Cricket_World_Cup>

How can we give this information to ChatGPT? The answer is quite simple. We just add this piece of text to our input query (as seen in figure 1.4).



##### Figure 1.4 ChatGPT response to the question, augmented with external context, Source : Screenshot by author of his account on [https://chat.openai.com](https://chat.openai.com/)

And there it is! ChatGPT, now, has responded with the correct answer. It was able to comprehend the piece of additional information we provided, distil the information about the winner of the tournament and respond with a precise and factually accurate answer.

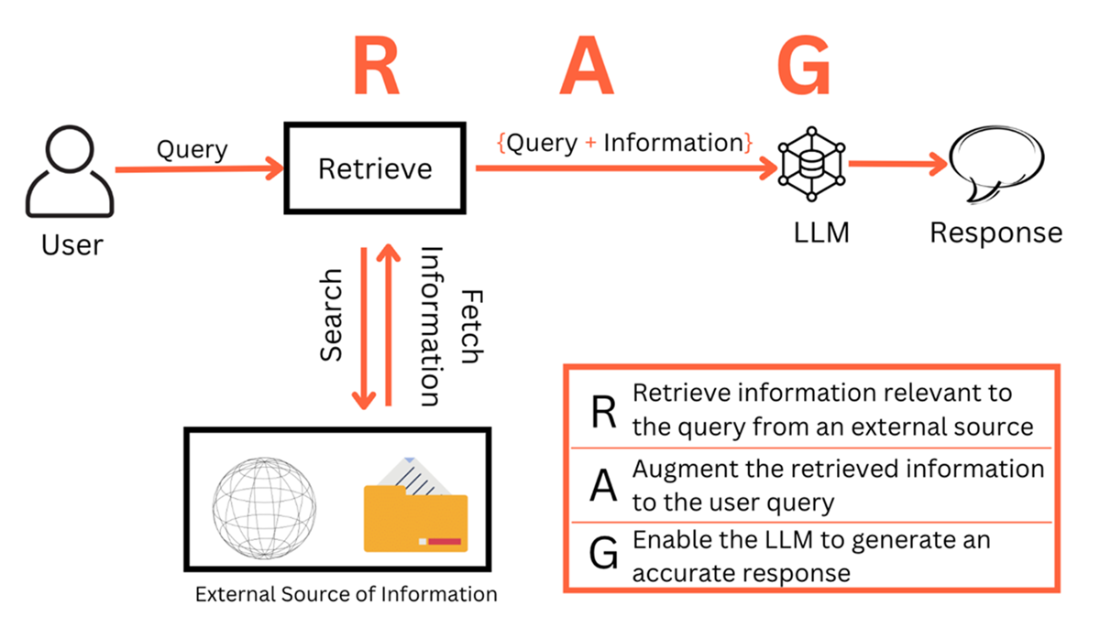
In an oversimplified manner, this example illustrates the basic concept of Retrieval Augmented Generation. Let us look back at what we did here. We understood that the question is about the winner of the 2023 Cricket World Cup. We searched for information about the question and identified Wikipedia as a source of information. We then copied that information and passed it onto ChatGPT, and the LLM powering it, along with the original question. In a way, we added to ChatGPT’s knowledge. Retrieval Augmented Generation, as a technique, does the same thing programmatically. It overcomes the limitations of LLMs by providing them with previously unknown information and, as a result, enhances the overall memory of the system.

As the name implies, Retrieval Augmented Generation, in three steps -

* **Retrieves** relevant information from a data source external to the LLMs (Wikipedia, in our example)
* **Augments** that external information as an input to the LLM
* Then, the LLM **Generates** a more accurate result.

A simple definition for RAG, also illustrated in figure 1.5 below, can therefore be as follows.

***“The technique of retrieving relevant information from an external source, augmenting that information as an input to the LLM, thereby enabling the LLM to generate an accurate response is called Retrieval Augmented Generation”***

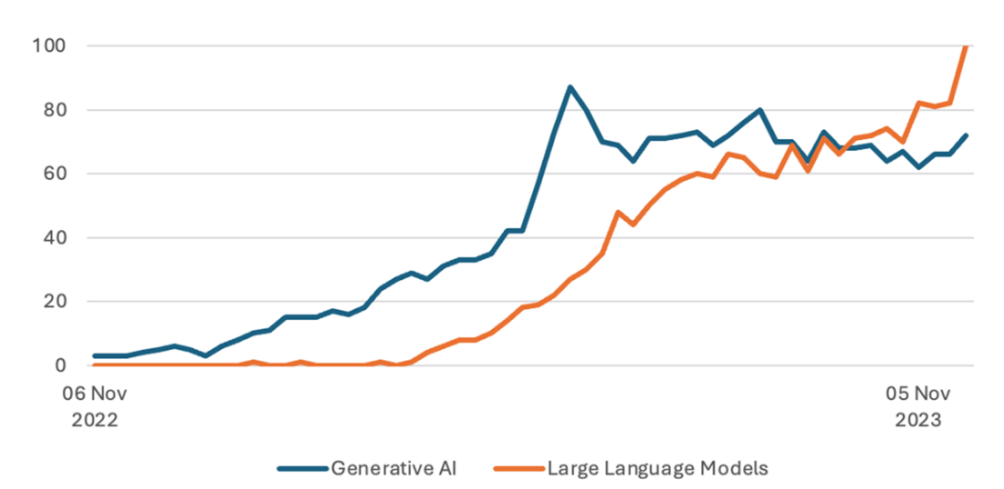
******

***Figure 1.5 Retrieval Augmented Generation: A Simple Definition***

The example that we have been looking at so far is an oversimplistic one. We manually searched for the external information and the search was for this one specific question only. In practice, all these processes are automated which allow the system to scale up to a diverse range of queries and data sources. This is what the subsequent chapters in the book will cover. But, before that, a brief understanding what LLMs are and how they can be leveraged will be helpful. We will understand what LLMs are, how they generate text and the concept of prompts. In case you are already familiar with these, you can skip this section and move to the next one.

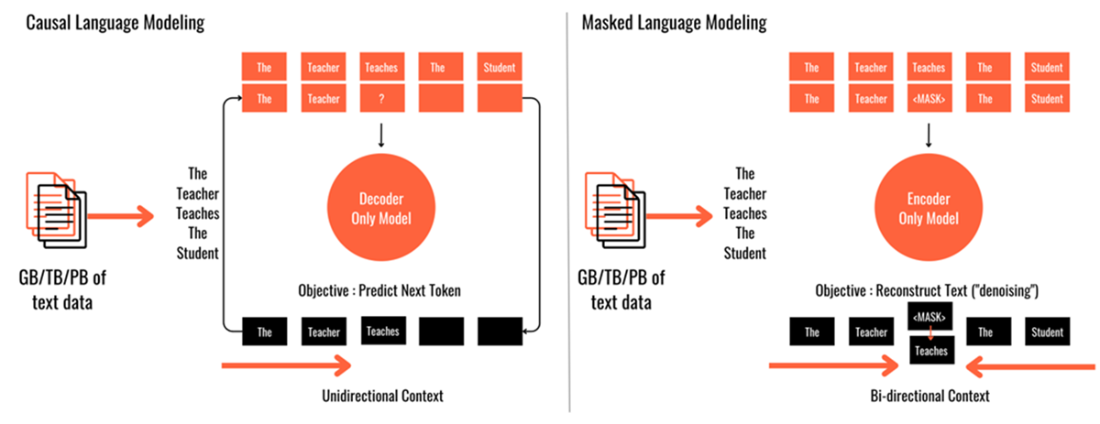
**1.2 What are Large Language Models?**

30th November 2022 will be remembered as a watershed moment in the field of artificial intelligence. OpenAI released ChatGPT and the world was mesmerized. Interest in previously obscure terms like Generative AI and Large Language Models (LLMs), skyrocketed over the following 12 months (as seen in figure 1.6).



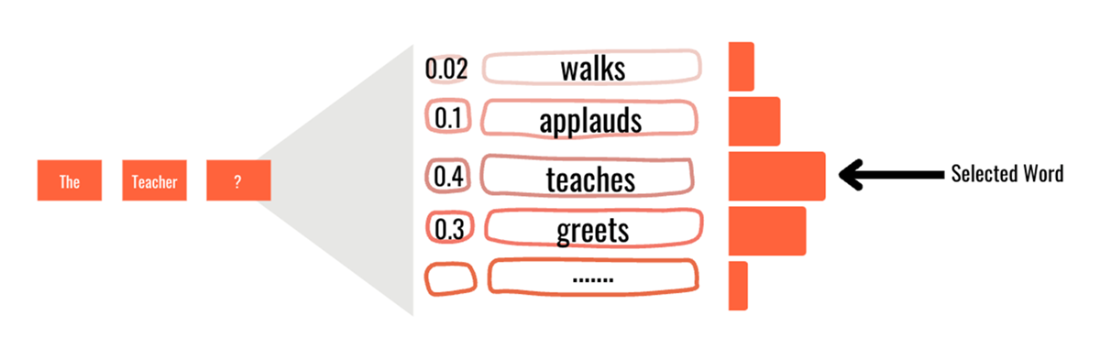
##### Figure 1.6 Google Trends of “Generative AI” and “Large Language Models” from Nov ’22 to Nov ‘23

Generative AI, and Large Language Models (LLMs) specifically, is a general-purpose technology that is useful for a variety of applications. LLMs can be, generally, thought of as a next token (loosely, next word) prediction model. They are machine learning models that have learned from massive datasets of human-generated text, finding statistical patterns to replicate human-like language abilities.

Very simplistically, think of the model first being shown a sentence like “The teacher teaches the student” for training. Then we hide the last few words of this sentence, “The teacher \_\_\_\_\_\_\_ and ask the model what the next word should be. The model should learn to predict “teaches” as the next word, “the” as the word after that and so on. There are various methods of teaching the model like causal language modeling (CLM), masked language modeling (MLM), etc. Figure 1.7 show the idea behind these two techniques. 

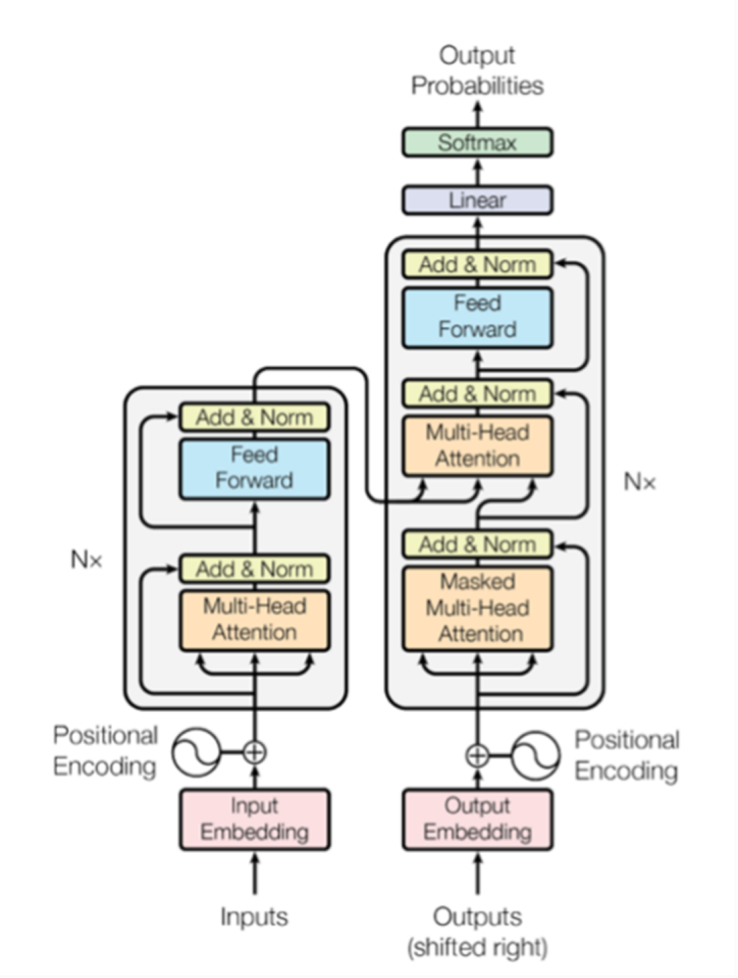
##### Figure 1.7 Two token prediction techniques – Causal Language Model & Masked Language Model

The training data can have billions of sentences of different kinds. The next token (or word) is chosen from a probability distribution observed in the training data. There are different means and method to choose the next token from the ones for which a probability has been calculated. In a crude manner, you can assume that a probability is calculated for all the words in the vocabulary and one amongst the high probability words is selected. For our example, “The teacher \_\_\_\_”, figure 1.8 shows an illustration of the probability distribution. The word “teaches” is selected because it has the highest probability. Other words could also have been selected.



##### Figure 1.8 Illustrative probability distribution of words after “The Teacher”

From a deeper technical perspective, Large Language Models have been made possible by a simple network architecture based on attention mechanism known as ‘transformers’. Prior to the introduction of transformers, tasks like language generation were accomplished using complex recurrent (RNNs) or convolutional neural networks (CNNs) in an encoder-decoder configuration. In their 2017 paper titled Attention Is All You Need (<https://arxiv.org/abs/1706.03762>), Vasvani et al, a part of the team at Google Research, introduced the transformers architecture and demonstrated remarkable efficacy in language translation tasks (Figure 1.9).



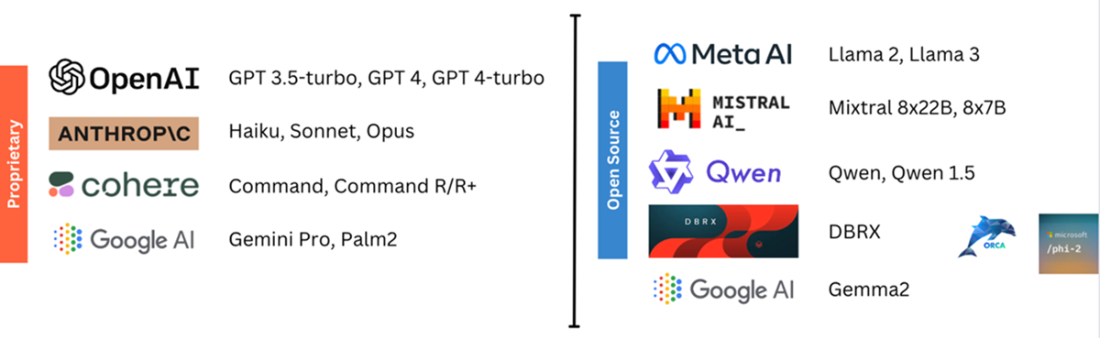
##### Figure 1.9 Transformer Architecture, Source: Attention is all you need, Vasvani et al.

The nuances of the transformers architecture and building LLMs from scratch is a wide area of study. In some use cases, building an LLM from scratch may be warranted but most applications rely on LLMs that have already been trained and available in the public domain. These models are called foundation models (or pretrained LLMs, or base models). They have been trained on trillions of words for weeks or months using extensive compute power.

WHEN IS TRAINING AN LLM FROM SCRATCH IS ADVISED?

Generally available foundation models are mostly trained in commonly understood language. Public data available on the open internet is one of the major sources of data. Therefore, if your use case is in a domain where the vocabulary and the syntax of the language is very different from commonly spoken language then chances are that the available LLMs may not yield optimal results. Domains like healthcare prescription data where the vocabulary is very specific or legal domain where the meaning of words is very different from common language may require collection of domain specific data and training a language model.

The GPT (GPT 3.5, GPT 4) series of LLMs by OpenAI, Claude 3 and it’s variants released by Anthropic, the Gemini series of models by Google AI, Command R/R+ by Cohere, as well as, open source models like Llama 2, Llama 3 by Meta AI, Mixtral by Mistral, Gemma 2, again, by Google AI are some popular foundation LLMs (as of April 2024) that are being used in a wide variety of AI powered applications.



##### Figure 1.10 Popular proprietary and open source LLMs as of April 2024 (non-exhaustive list)

WHAT ARE MODEL PARAMETERS?

You may have heard that Large Language Models have billions and trillions of parameters. GPT-4 has 1.76 trillion parameters. Meta’s Llama models come in three different sizes and are denoted as 7B, 13B and 70B models. These are nothing but the number of parameters. So, what exactly are parameters? All machine learning models including LLMs are mathematical models of the form y=f(x) where y is the model and x are the features of the training data. Imagine an equation where y= w + b1x1 + b2x2 + b3x3 +…. + bnXn. Here w, b1, b2,…bn are the values that the model adjusts or learns during training. These values are called model parameters. The larger the number of parameters, the bigger is the size of the model and the more computational resources are required. On the other hand, a large sized model is expected to perform better.

Large Language Models have found applicability in a wide variety of tasks because of their language understanding and text generation capabilities. Some of the application areas are -

Writing – Generating pieces of content like blogs, reports, articles, posts, tweets etc.

Summarization - Shortening long documents into a meaningful shorter length.

Language Translation - Translating text from one language to the other.

Code Generation- Writing code in a programming language given certain instructions.

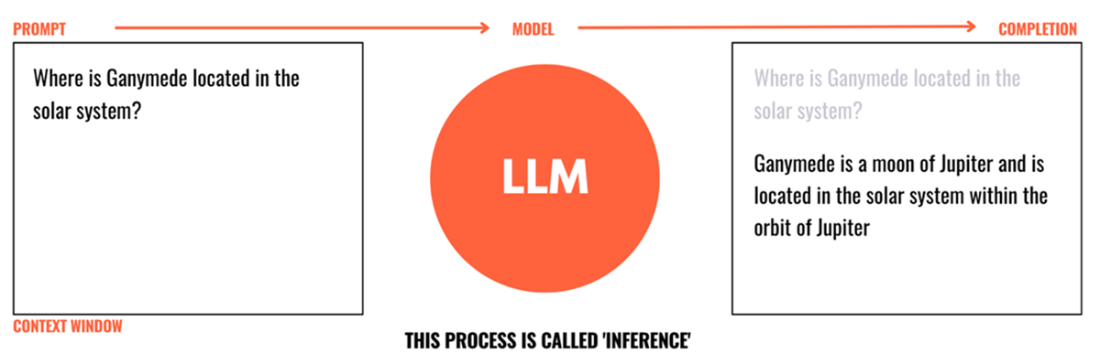
Information Retrieval - Extract specific information from text like names, locations, sentiment.

Classification – Classifying pieces of text into groups.

Conversations- Like question answering or chat.

LLMs is a rapidly evolving technology. Learning about LLMs and their architecture is a large area of study. Since, this book focusses on leveraging Retrieval Augmented Generation to use the available LLMs we will, therefore, not delve deep into the transformer architecture and the LLM pre-training process. We will, instead, spend some time in knowing how one interacts with the already available pretrained LLMs.

1.2.1 How do you work with Large Language Models?

Interacting with LLMs differs from traditional programming paradigms. Instead of formalized code syntax, you provide natural language (English, French, Hindi, etc.) input to the models. ChatGPT, as a widely popular example of an LLM powered application, demonstrates this. These inputs are called “prompts”. When you pass a prompt to the model, it predicts the next words and generates an output. This output is termed a “completion”. This entire process of passing a prompt to the LLM and receiving a completion is known as “inference”. Figure 1.11 illustrates the inferencing process.

##### Figure 1.11 Prompt, Completion, and Inference

Prompting an LLM may, at the first glance, seem like a simple task since the medium of prompting is commonly understood language like English. However, there’s more nuance to prompting. The discipline of constructing effective prompts is called prompt engineering. Practitioners and researchers have discovered certain aspects of a prompt that assist in getting better responses from an LLM. For example –

Defining a “Role” for the LLM like “You are a marketer who excels at creating digital marketing campaigns”, or “You are a software engineer who is an expert in python” has been demonstrated to increase the quality of responses.

Giving “examples” within the prompt has emerged to be one of the most effective techniques to guide the LLM responses. This is also known as Few Shot Learning (FSL).

It has also been observed that giving clear and detailed instructions helps in the adherence to the prompt.

Prompt engineering is an area of active research. Several nuanced prompting methodologies discovered by researchers have demonstrated the ability of LLMs to handle complex tasks. Chain Of Thought (CoT) prompting, Reason and Act (ReAct), Tree of Thought (ToT) and more prompt engineering frameworks are witnessing their use in several AI powered applications. We will refrain from going deeper into the discipline of prompt engineering right now but look at it, in the context of RAG, in chapter 4. However, an understanding of a few basic terms with respect to LLMs will be beneficial.

Context Window: The nature of the underlying architecture puts a limit on the number of tokens that can be passed to the LLM. This limit is called the Context Window. It is a critical component of LLM usage as it will restrict the amount information that can be passed to the model and the amount of words the model will generate.

Temperature: LLM outputs are based on the probability of generated token. Temperature controls the randomness of generation. The higher the temperature, the more random the output will be.

Few Shot Learning: LLMs have been observed to perform better if they are provided with certain examples of the desired output within the prompt. This technique of providing inputs is called Few Shot Learning.

In-context Learning: During inference, passing a prompt to an LLM does not alter the underlying model’s memory. The model in its responses may still take into account the information that has been augmented in the prompt. This process, in which the model learns new information without changing any of the underlying parameters, is also called in-context learning.

Bias and Toxicity: LLMs are trained on huge volumes of unstructured data. This data comes from various sources (predominantly, the open internet). The model may show favoritism or generate harmful content based on this training data.

Supervised Fine Tuning (SFT): For some tasks, prompt engineering alone does not yield satisfactory results. In such cases, the model is further trained by providing it with a set of examples. As opposed to in-context learning. This process changes the weights of the underlying model and consequently alters the memory of the model to suit the task at hand. This process is called Supervised Fine Tuning.

Small Language Models (SLMs): SLMs are like LLMs but with lesser number of trained parameters (therefore called "Small"). They are faster, require less memory and compute, but are not as adaptable and extensible as an LLM. Therefore used for very specific tasks.

The domain of LLMs is expansive and an area of study in itself. You will come across the concepts like Reinforcement Learning from Human Feedback (RLHF), Parameter Efficient Fine Tuning (PEFT), various model deployment and monitoring techniques. We will discuss these concepts in the context of RAG in upcoming chapters.

LLMs have really captured the imagination of both researchers and practitioners. The world is now largely aware of the massive ability the LLMs hold. But, as is the case with any technology, LLMs also have their own set of limitations. While we touched upon these limitations in the first section, let us look at them in more detail and set the stage up for a deeper exploration of Retrieval Augmented Generation.

1.3 The Curse of the LLMs and the novelty of RAG

We discussed previously how ChatGPT got quite popular very soon. It became the fastest app ever to reach a million users. The usage exploded in a matter of days and, so did the expectations. Many users started using ChatGPT as a source of information, like an alternative to Google Search. They looked at LLMs for knowledge and wisdom, yet LLMs, as we now know, are just sophisticated predictors of what word comes next.

As a result, the users also started encountering prominent weaknesses of these models. There were questions around copyright, privacy, security, etc. But people also experienced the more concerning limitations of Large Language Models that raised questions around the general adoption and value of the technology.

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 always up to the current. e.g. The latest GPT4 Turbo model released by OpenAI on 9th April, 2024 has knowledge only till December 2023. Any event that happened after this knowledge cut-off date, the information of that event is not available to the model.

Hallucinations

Often, it is observed that LLMs provide responses that are factually incorrect (We saw this in the 2023 Cricket World Cup example at the beginning of this chapter). Despite being factually incorrect, the LLM responses sound extremely confident and legitimate. This characteristic of “lying with confidence”, called hallucinations, has proved to be one of the biggest criticisms of LLMs.

Knowledge Limitation

LLMs, as we already read, have been trained on large volumes of data sourced from a variety of sources including the open internet. They do not have any knowledge of information that is not public. The LLMs have not been trained on non-public information like internal company documents, customer information, product documents, etc. So, LLMs cannot be expected to respond to any query about them.

These limitations are inherent to the nature of LLMs and their training process. While the weaknesses of LLMs were being discussed, a parallel discourse around providing additional context or knowledge to the LLMs models started. In essence, it meant creating a ChatGPT like system for proprietary or non-public data with three main objectives.

Make LLMs respond with up-to-date information.

Make LLMs respond with factually accurate information.

Make LLMs aware of proprietary information.

These objectives can be achieved through diverse techniques. A new LLM can be pre-trained from scratch that includes the new data. An existing model can also be fine-tuned with additional data. However, both the approaches require significant amount of data and computation resources. Also, updating the model at a regular frequency with new information is equally costly. In majority of the use-cases, these costs turn out to be prohibitive. Enter, Retrieval Augmented Generation, a cheaper, more effective and dynamic technique to attain the three objectives.

1.3.1 The Discovery of Retrieval Augmented Generation

In May 2020, Lewis et al in their paper, **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks** (https://arxiv.org/abs/2005.11401), explored the recipe for RAG - models which combine pre-trained ‘parametric and ‘non-parametric’ memory for language generation. Let us pay some attention to these terms ‘parametric’ and ‘non-parametric’.

Parameters in machine learning parlance refer to the model weights or variables that the model learns during the training process. In simple terms, they are settings or configurations that the model adjusts in order to perform the assigned task. For language generation, LLMs are trained with billions of parameters (GPT 4 model has 1.76 trillion parameters and the largest Llama 3 model has 80 billion parameters). 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.

Conversely, we can provide information to an LLM that it does not have in its parametric memory. We saw in the example of the Cricket World Cup that when we provided information from an external source to ChatGPT, it was able to get rid of the hallucination. This information that is external to the LLM, but can be provided to the LLM is termed “non-parametric”. If we can gather information from external sources as and when desired and use it with the LLM, it forms the “non-parametric” memory of the system. In the aforementioned paper, Lewis et al, stored Wikipedia data and used a retriever to access the information. They demonstrated that this RAG approach outperformed parametric-only baseline in generating more specific, diverse and factual language. We will discuss vector databases and retrievers in chapter 3 and chapter 4.

In 2024, RAG has become one of the most used technique in the domain of Large Language Models. With the addition of a “non-parametric” memory, the LLM responses are more grounded and factual. Let us discuss the advantages of RAG.

1.3.2 How does RAG help?

With the introduction of ‘non-parametric’ memory, the LLM does not remain limited to its internal knowledge. We can, at least theoretically, conclude that this non-parametric memory can be extended as much as we want. It can store any volume of proprietary documents or data and have access to all sorts of sources like the intranet and the open internet. In a way, through RAG, we open up the possibility of embellishing the LLM with unlimited knowledge. There will always be some effort required to create this non-parametric memory or the knowledge base and we will look at it in detail later. Chapter 3 in this book is dedicated to the creation of the non-parametric knowledge base.

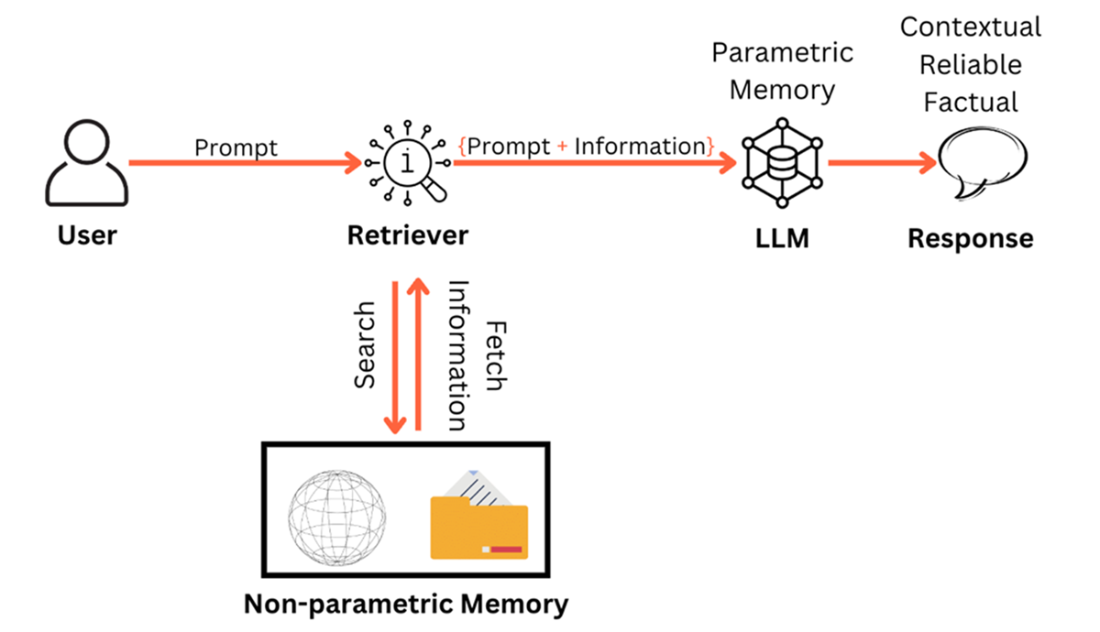
As a consequence of overcoming the challenge of limited parametric memory, RAG also builds user confidence in the LLM responses.

The added information assists the LLM in generating responses that are contextually appropriate and the users can be relatively more assured. For example, if the non-parametric memory contains information about a particular company’s products, users can be assured that the LLM will generate responses about those products from the provided sources and not from elsewhere.

* In addition to being context aware, because the information is being fetched from a known source, these sources can be cited in the response. This makes the responses more reliable since the users have the choice of validating the information from the source.
* With contextual awareness, the tendency of LLM responses to be factually inaccurate is greatly reduced. The LLMs hallucinate less in RAG enabled systems.
* We started with a simple definition for RAG at the beginning of this chapter. Let us now try and expand that definition.

The technique of enhancing the parametric memory of an LLM by creating access to an explicit non-parametric memory, from which a retriever can fetch relevant information, augment that information to the prompt, pass the prompt to an LLM to enable the LLM to generate a response that is contextual, reliable, and factually accurate is called Retrieval Augmented Generation

This definition is illustrated in the figure 1.12 below.



##### Figure 1.12 RAG enhances the parametric memory of an LLM by creating access to non-parametric memory

Retrieval Augmented Generation has acted as a catalyst in the propagation and acceptance of LLM powered applications. Before concluding this chapter and getting into the design of RAG enabled systems, let us look at some popular use cases where RAG is being adopted.

1.4 Popular RAG use cases

RAG is not just a theoretical concept but a technique that is as popular as the LLM technology itself. Software developers started leveraging language models as soon as Google released BERT in 2018. Today, there are thousands of applications that leverage LLMs to solve language intensive tasks. Whenever you come across an application using LLMs, more often than naught, it will have an internal RAG system in some shape and form. Common applications include –

Search Engine Experience: 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.

Personalized Marketing Content Generation: The widest use of LLMs has probably been in content generation. Using RAG, the content can be personalized 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.

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 customized 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 specialized agents depending on the nature of the query. Almost all LLM based chatbots on websites or as internal tools use RAG.

Document Question Answering Systems: As we have discussed, one of the limitations of LLMs is that they don’t have access to proprietary non-public information like product documents, customer profiles etc. specific to an organization. With access to such proprietary documents, a RAG enabled system becomes an intelligent AI system that can answer all questions about the organization.

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 behavior using RAG, these assistants are set to become more personalized.

AI powered research: AI agents are gaining traction in research intensive fields like law and finance. RAG is being extensively used to retrieve and analyze case law to assist lawyers. A lot of portfolio management companies are introducing RAG enabled systems to analyze scores of documents to research investment opportunities. RAG is also being employed for ESG research.

This introductory chapter dealt with the concept of Retrieval Augmented Generation. We also got a brief overview of Large Language Models and how one interacts with them. Overcoming the limitations of LLMs, RAG addresses these challenges by providing access to a non-parametric knowledge base to the system. Finally, we looked at some use cases of RAG.

With this foundational understanding of RAG, in the next chapter we will take the first step towards understanding how RAG enabled systems are built by looking at the different components of their design.

1.5 Summary

RAG enhances the memory of LLMs by creating access to external information.

LLMs are next word, (or token) prediction models that have been trained on massive amounts of text data to generate human-like text.

Interaction with LLMs is carried out using natural language prompts and prompt engineering is an important discipline.

LLMs face challenges of having a knowledge cut-off date and being trained only on public data. They are also prone to generating factually incorrect information (hallucinations).

RAG overcomes the limitations of the LLMs by incorporating non-parametric memory and increases the context awareness and reliability in the responses.

Popular use cases of RAG are search engines, document question answering systems, conversational agents, personalized content generation, virtual assistants among others.

2 RAG-enabled Systems and Their Design

This chapter covers

Concept & Design of a RAG-enabled system

Overview of the Indexing Pipeline

Overview of the Generation Pipeline

Overview of RAG Evaluation

Overview of LLMOps Service Infrastructure

In the previous chapter, we explored the core principles behind Retrieval Augmented Generation and the challenges faced by Large Language Models that RAG addresses. To construct a RAG-enabled system there are several components that need to be assembled. This includes creation and maintenance of the non-parametric memory, or a knowledge base, for the system. The other pipeline facilitates real-time interaction by sending the prompts to and accepting the response from the LLM, with retrieval and augmentation steps in the middle. Evaluation is yet another critical component, ensuring the effectiveness of the system. All these components of the system are supported by a robust service infrastructure.

Jn ayjr trcpeah, vw jwff iatdel rxp gineds le c CRN-laebden mtssye, nigixaenm prk etpss edvonliv nsy yrk knqv lte xwr inderfeft ipeslinpe. Mo fjwf sfzf vrd niipepel rucr estacre rkp engloewkd pzav sc rbv nigxiend ilepniep. Cux hetor epelnipi rzgr oalswl ftxs rkjm ietoanircnt bwrj rxu EVW jfwf xq dfrrreee rk az xdr ogneiteran ipeipnle. Mx fjfw iusdcss our divnadiuli nstencopmo vl seeth jokf rgzz gldoain, gdemnesbdi, ctveor trseso, rverritees uzn vtvm. Yiylaodinltd, ow jwff ykr cn dgndnatursien el wbe rqv loiaeuavnt lx TYK bledane estssmy jz tcenocudd. nsb tirnducoe roy eevscri tteirrcasruunf rrzb rspoew ayua tsemyss.

Rbaj htrcepa jc nz iitnrtcuodno rx sraouiv encsptonom yzrr ffjw oq ssdcsuedi jn tdaeil nj brk icnomg rpstecah. Ru rqk qnv kl pcrj tprehca, deq jwff kvsy aecruqdi c ohuk dnatnusigendr le rvq ooncsptemn lx c ARQ-dnaebel mstsye nsh gk aeyrd kr vqxp kujx jnrx rkd frenditfe cpeoonmsnt.

By the end of the chapter you should -

Xk vqzf re tnrnuedsda rgv eesalvr monpocntse xl rbv BBU yemtss digens

Srk fuyloesr dd lkt z eepedr prnolixoeta lk vbr diinngex eieilppn, rkd ntaiogener inpselpei, YCD niuavlatoe soedthm ysn obr esirecv unrtrrctafsuei.

2.1 RAG-enabled Systems

Yu wen, wv oxds mvkz rv knwv rprz AYK jz z ivtla omencntpo xl rxg esysmst rzdr laeevger Exctq Faaggneu Wedslo vr ovles ithre qoa asces. Ary, ywk aebo aggc s ssyetm xkvf fkjo? Xv ltutrislea bzjr, wv jffw triivse rux lxaempe ow hoga cr rdx ingnngeib kl Yeratph 1 (“Mxb xnw rdo 2023 Tcrtkei Mpfxt Adp?”) ngc fhz xrd rob tspse wv dokntroue re nlbeea XgsrKLB rv oeivdpr dc rjpw uro caaeutrc reposnse.

Adk ialinti rabk wzs agnisk uro iunoqste efltsi - “Meu vnw uro 2023 Becitkr Mufte Byh?”. Vgonlliow ayrj, wx ayanumll seacrehd tlx souecsr ne urv teirtnen chiwh hmtgi dkos oanirfmiotn gnirerdga rod saenrw rx qxr eitqosun. Mo onfud knk (Mipiekaid, nj gtk aelpmxe) sny dcxtretae z eavlernt arhppgaar lktm rbo rocues - Yvq rnttodrouyic hapagarpr lk ukr Mieipakid lecrati nx 2023 Xreikct NUJ Mtfvy Buq. Sbqynlueetus, wk addde yrx envlreta pgrpahara er yet nloigari itsqueno, pdssea oru ntueosiq hns our retedriev hpgarpraa, egthetro, jn odr pmptro re BsurNVR uzn reb z lfaualyct rtcreoc prnsoees – “Bauslaitr vnw kbr 2023 Betikcr Mgtef Aqb”.

Cjcb nsa kq dtlsdliie rejn z xojl-rkqc rescsop. Ktb tsmesy enesd xr citalteiaf ffz yor ovjl spset.

Sgxr 1: Dzxt ezcc s tiouenqs re yte tmeyss

Sgkr 2: Cdx emssyt eascserh let aofntinmiro nlvrteae xr rgk ipnut teuisqon

Shro 3: Yod foiantroinm leeravnt xr brx iutpn iounseqt jc etecdhf, tk evrritede, cpn dddae kr obr pitun uqseniot

Skrq 4: Cjqc enitqosu + nnoitfrioam ja psesda re ns PFW

Sodr 5: Rkd FPW edprssno wjrp c ontacetlux seanrw

Jl eug lrelca, kw ocod dreayal wandr rkq crbj pcseosr nj Xthaerp 1. Frk qz lizivusae rj nj orb oxnctte lk heste jvkl petss cz nwsho nj rvp frieug 2.1 wlebo. Ckq veaob wkoowrfl jffw dx ldacel urk Ktieannroe Eieilnep ecins rj ja ruk knk nringgaeet drk nrwsea.