GenAI Pinnacle Program.

Case Study 1 :-

1	2	3
Goat or Ferrari	Goat or Ferrari.	Goat or Ferrari.

Person to Select

* Reject Inference.

---

* Unimodal Foundational Models
  ↳ The models that are trained for or to deal with one class of dataset.
  Like text, Image etc.
  e.g GPT-3, Llama-2, PaLM-2 are the example of unimodal foundation model.

* ~~Note~~ ↳ Diffusion models are the unimodal foundation model that deals with image dataset.

* Multimodal Foundation Models
  ↳ Trained on more than one class of dataset like text to image, audio-video etc.

- Prompt Engineering:- (Part 1).

i) Chain-of-thought prompting is where we give model an example which has steps explained for the reasoning help in model to work like that.

ii) Self-consistency prompting is a method that helps language models to solve complex problems by ~~solving~~ considering multiple ways to think about a problem and then choosing the most common solution.

iii) Verify-and-edit-prompting is a method that enhances answer accuracy by using external sources to verify and refine respon- e.g using wikipedia.

iv) Tree-of-thoughts prompting is a method to solve a problems by exploring different paths or 'thoughts' that lead to a solution. It work like human where we try to explore multiple path before reaching to one.

v) Graph of thoughts prompting is a method to help LLM organise information in a network, with each data point connected, mirroring neural pathways.

vi) ReAct prompting: It is a method that combines thinking and action, boosting language models problem-solving abilities. (Reason + Act).

Skeleton of thoughts prompting is a prompting method that outline answer first and then fills in details in parallel, speeding up the response.

Rephrase and respond prompting :- It would help the user to rephrase the question and then respond.

Self-Refine prompting :- It is an iterative process where a language model autonomously generates, assesses and refines answers.

Chain of Natural language Inference prompting is a hierarchical framework designed to address and reduce hallucinations in text generated by large language models (LLMs).

It is divided into Detection agent where the sentence is ~~divide~~ divided into seperate sentence and then additional source document is uploaded to check if ~~it~~ the sentence is hallucination or not. If sentence is identified as hallucinated then we check one more time with entity level detection. The hallucinated sentences pass through chain-of-thoughts. Then we use this reasoning along with ~~generated text~~ source text and mitigation instruction the remove halluci

Chain-of verification prompting :- helps LLM to self-check and refine their responses through a series of validation question.

→ GPT-4 are the example of multimodal foundation model combining text and image data.

→ Stable diffusion, DALL-E 3 are also an multimodal foundation model for text and image.

→ AudioGen and AudioCraft are the multimodal model for audio.

* Foundational Model :-
The model that are trained on large set of unlabeled dataset like image, audio, text and which can be ~~fout~~ further tuned to perform specific task.

* Different types of LLM.
  ↳ Based on response. → (Instruction following LLM)
  ↳ Based on model architecture.
    * Encoder Based
    * Decorder Based.
    * Encoder and Decoder Based LLM.

* Decooder LLM
  ↳ GPT-3.
  ↳ GPT-4
  ↳ Llama-2
  ↳ Falcon LLM

* Encoder and Decoder
  ↳ T5
  ↳ Flan T5
  ↳ Switch
  ↳ MT5

* So for Decoder based LLM shows the excellent result.

* 4 different ways to build LLM Application.
  - ↳ Prompt Engineering.
  - ↳ Retrieval Augmented Generation (RAG)
  - ↳ Finetuning LLM
  - ↳ Training LLM from stratch.

i) <u>Prompt Engineering</u>:-

Prompt is a text given to LLM to get answer we want. Why it is important because LLM can make mistakes and fails to provide correct output.
It helps in improving performance and helps in achieving desired output.

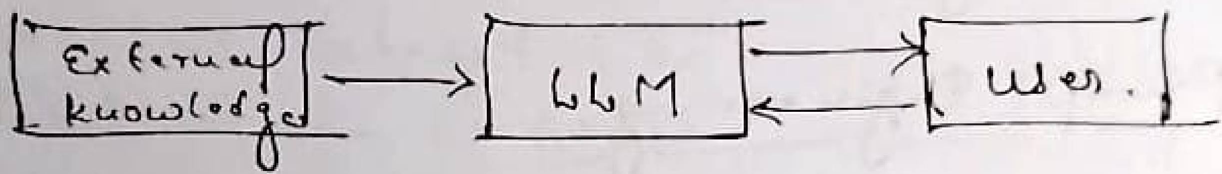
* Pros:-
  - → No technical req.
  - → No training req.
  - → No training data.
  - → No compute resource.
  - → Very minimal cost.

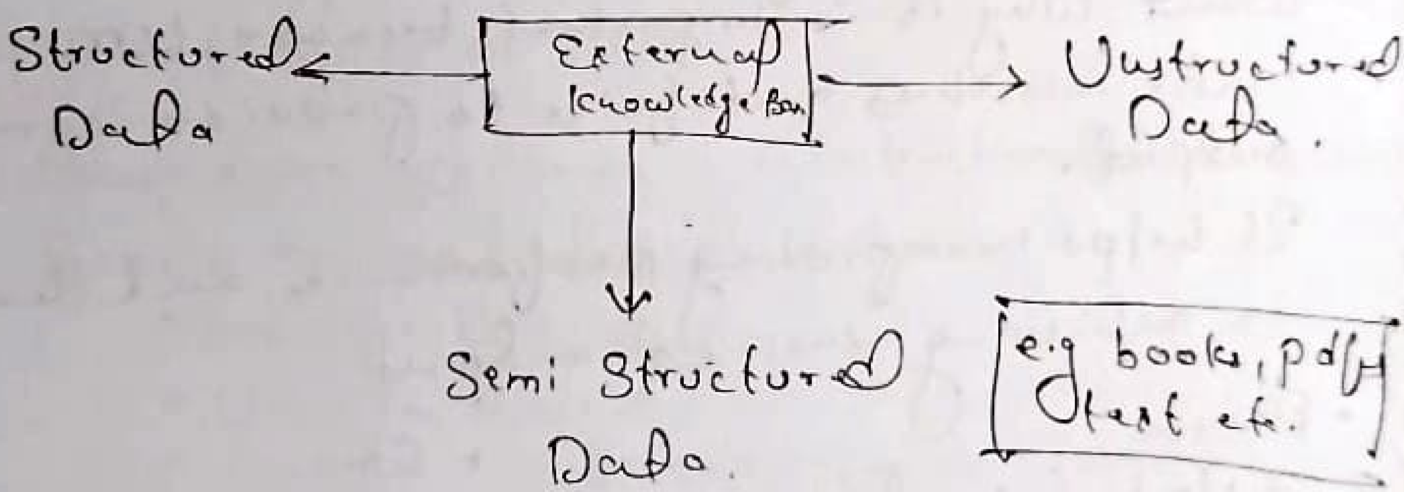
* Cons.
  - → Inconsistent.
  - → Hallucination.
  - → Very less info. fit.

* Retrieval Augmented Generation.

→ Why RAG :- As we know the knowledge of ChatGPT is not upto date and it won't be able to give info" post the date it was trained.

→ with the help of RAG we can ingest the external knowledge into the LLM and get the response using prompt.





→ Once external data connected to LLM we can create QA system to ask questions regarding our data.

→ Build chatbot on your enterprise data.

→ Building RAG application doesn't involve any model training.

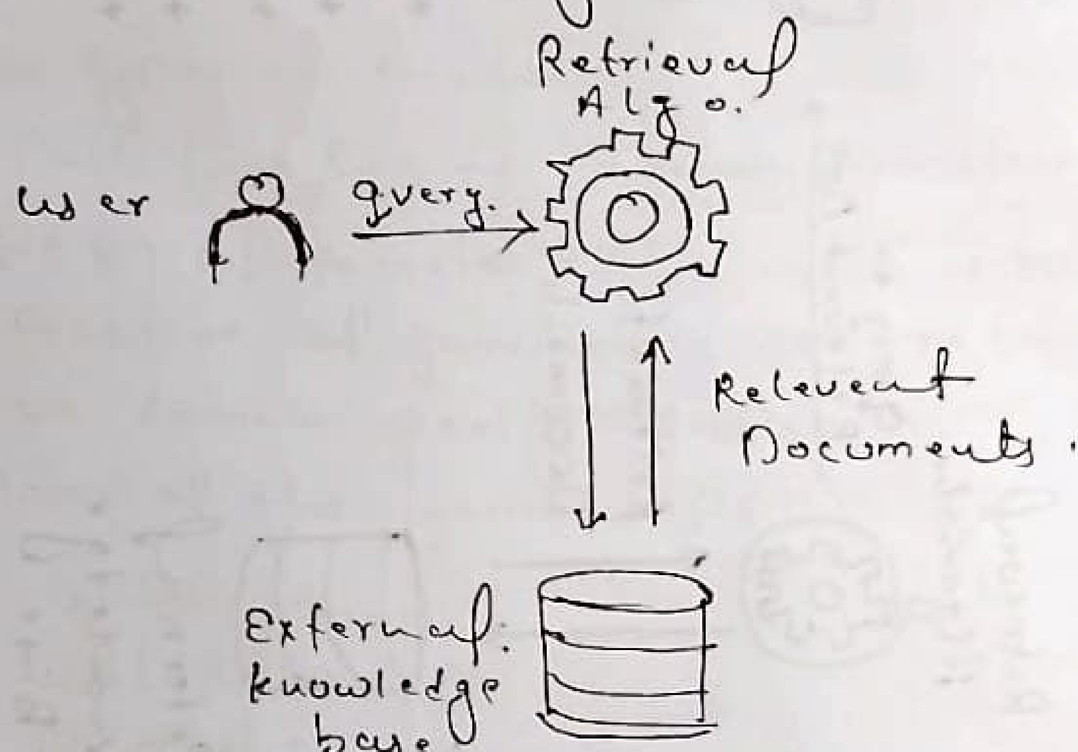
How does RAG works.

Two main component of RAG.

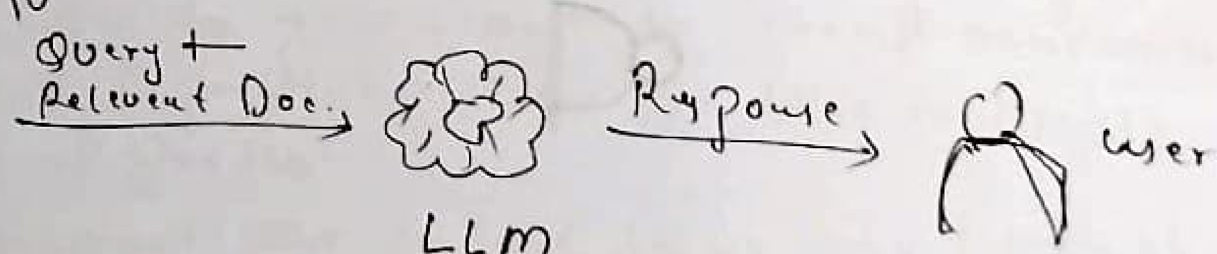
i) Retrieval

ii) Generatory..

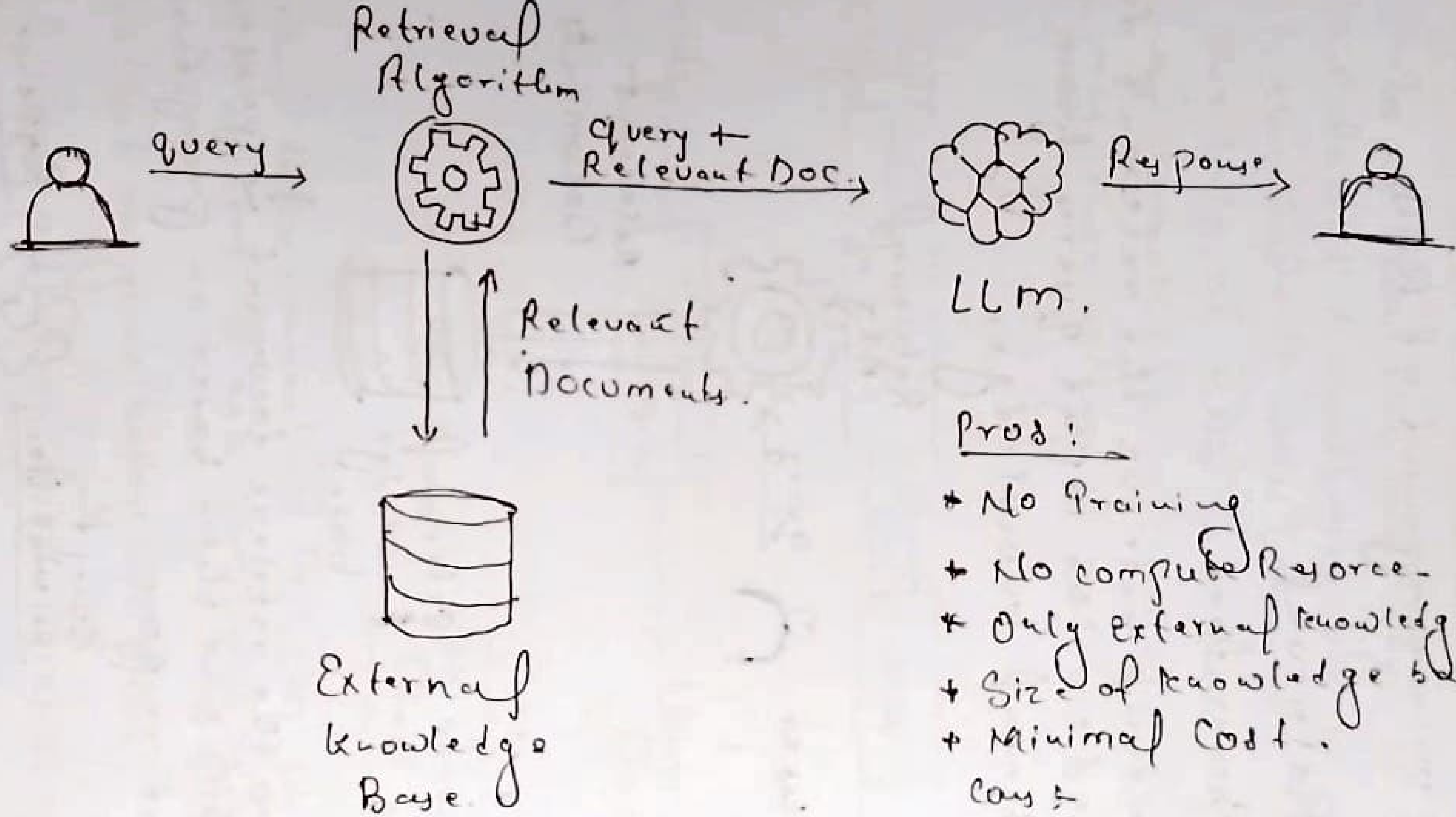
Retrieval retrieve the relevant documents based on the input query from ~~know~~ External knowledge.



Now the retrieve document is passed to the LLM and then based on prompt user can refrim the respons-



Complete Architecture of RAG



Pros :

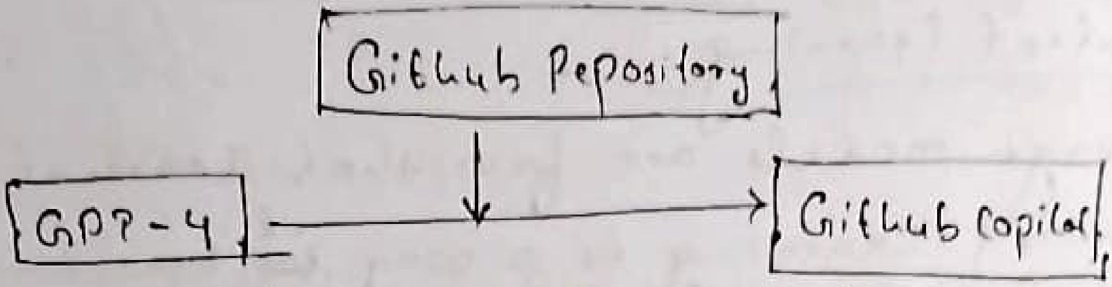
+ No Training
+ No compute Resorce.
+ Only external knowledge.
+ Size of knowledge base
+ Minimal Cost.

Cons :

+ Limited external knowledge.
+ Wrong output.
+ Hallucination

Finetuning LLM.

Finetuning is required as to adapt the LLM to perform domain specific task.



Pretrained Model

Fine tunned Model.

* Two types of fine tuning
  i) Full fine tuning. → ~~Retrain~~ Retrain. parameters of entire model
  ii) PEFT (Parameter Efficient Fine tuning).
  Fraction of parameters are retrained on the domain specific data.
  Some of the famous PEFT
    LORA –
    AdaLoRA
    QLoRA.

* Temperature Parameter:-
- → The use of temperature parameter tells your model how do you want to treat randomness.
- → When model predicts there are like multiple options of prediction.
- → If you want the output to be very deterministic set temp to low.

- GPT-3 are optimized for text completion task and are hence called autoregressor models. They are not trained to perform task apart from this.

* In Context Learning :-

- Language models are few shot learner.
- In context learning is a way of learning to a model where the input to the model with set of task with some example.
- Then model would be able to perform on new task it has just learned from in context with examples.

- Types of In-context learning :-
  i) Few Shot Prompting
  ii) One shot Prompting
  iii) Zero Shot Prompting.

i) Few shot prompting can be performed by providing few set of examples.

ii) In one shot the model only see one example with task description.

iii) Zero Shot is without any example.

* With the help of In-context learning we can ~~pro~~ protect the enterprise data. However the maximum length is set for in-context learning and it would ask you to reduce the token. Few tokens limit example are 8k, 16k, 32k.

* However this limitation can be overcome by breaking the entire context into multiple context and in that case based on user ~~con~~ question it ~~fect~~ fetches relevent chunk to answer the question.

* However when we deal with huge data say GBs PBs in that case context ~~fase~~ based on chunks fails and in that case ~~it~~ RAG would come into picture.

RAG :-

Retrieval Augumented Re Generation is a technique that allow the LLM powered system to connect with custom data.

The external knowledge can be :-

→ PPt

→ Raw files

→ Audio files.

→ Video files

→ Database

→ API.

* RAG is a way to ~~sp.~~ work on your enterprise data.

* Steps in RAG.

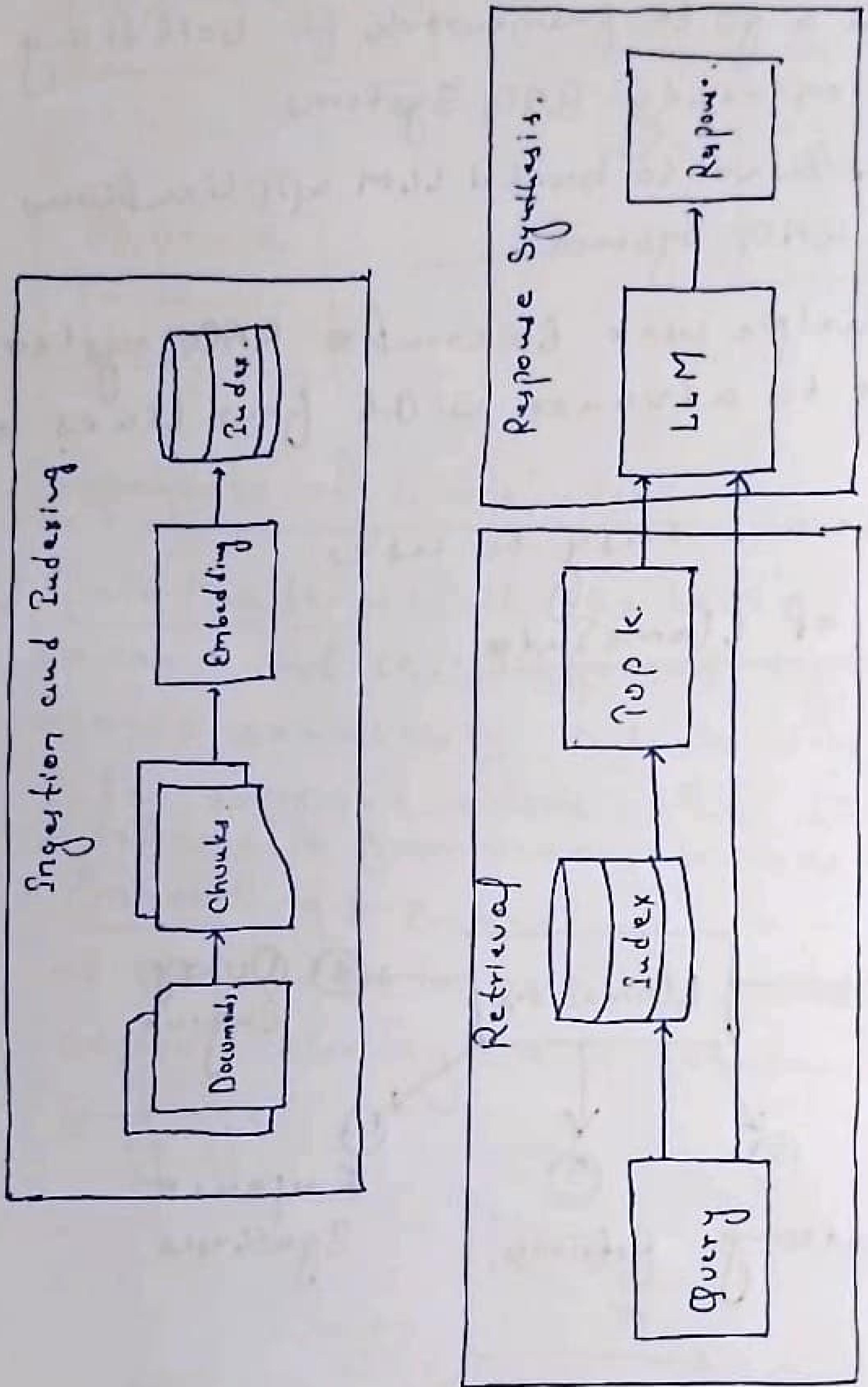
Data Ingestion → Indexing → Retrieval → Response Synthesis

↓

Evaluation ← Query

- Data Ingestion where the external knowledge prepared in the form ~~of~~ which is required for RAG.
- Indexing where documents are split into chunks and created embedding.

RAG framework.

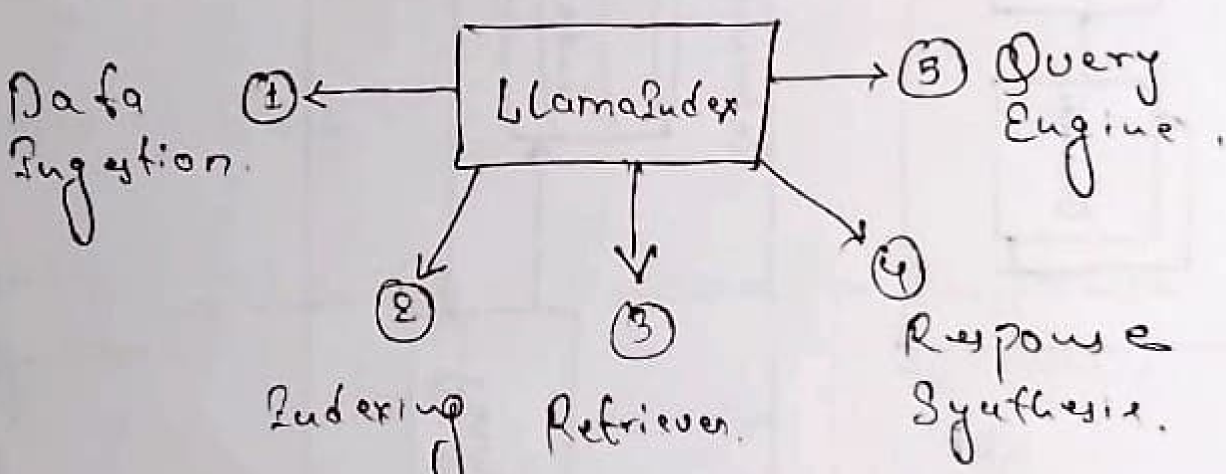


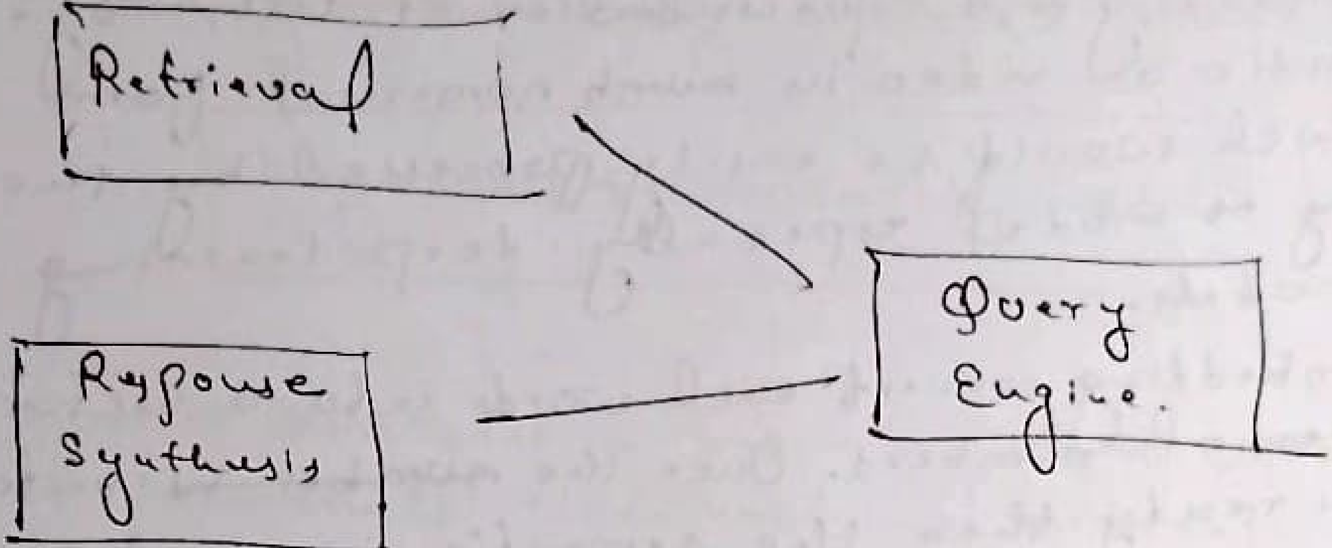
* Requirement for RAG.
  - → framework to implement RAG.
  - → Embedding
  - → Vector Store.
    - • Store vector embedding.
  - → LLM

* LlamaIndex :-

- LlamaIndex is a go to framework for building the production ready RAG systems.
- LlamaIndex is used to build LLM applications powered by RAG System.
- LlamaIndex helps user to create RAG system from simple to advance with few lines of code.
- It is simple and easy to use.

* Components of LlamaIndex.

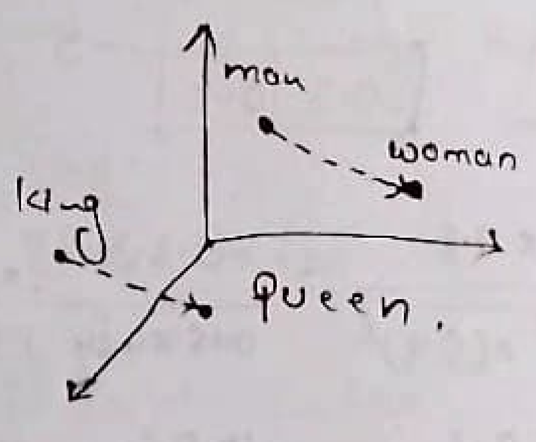




* __Components of LlamaIndex:__

i) __Data Loader__:- With the help of data loaders we can read the different types of data source seamlessly. Data loaders from Llama-Index support more the 100 types.
It takes data from various sources in document format

ii) __Embedding__:- Embedding is a representation of text data in numeric format. Embedding capture the symantic relationship in the language.



→ Embedding is a representation of text, image audio or video in ~~numb~~ numerical form which would be easily processed by the ~~sy~~ ~~tt~~ model especially deep learning models.

→ Embedding convert each words into a vector ~~rer~~ of numbers. Once the ~~number~~ is vector is ready then the semantic is calculated with the help of cosine similarity.

$$S_c(A, B) = \cos\theta = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

$$\boxed{= \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2 \cdot \sum_{i=1}^{n} B_i^2}}}$$

e.g

Person 1		Person 2	
-0.4	0.8	-0.3	0.2

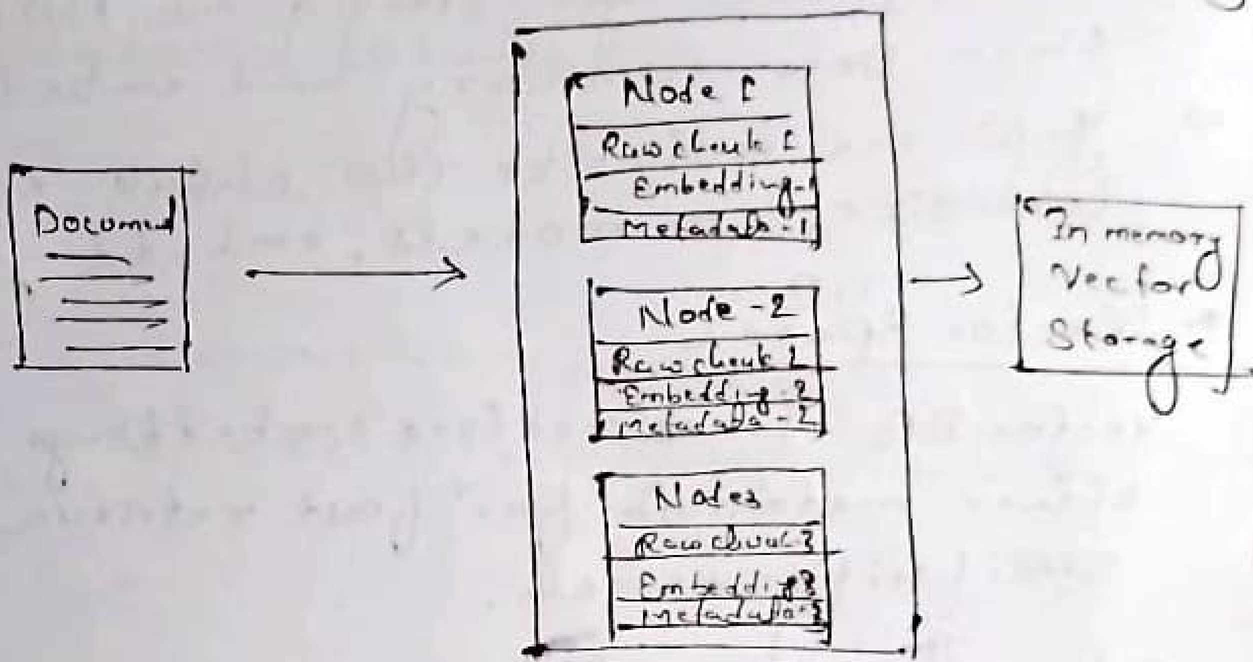
$$= \frac{-0.4 \times 0.8}{\sqrt{(-0.4)^2 \times (0.8)^2}} = \frac{-0.32}{0.16 \times 0.64} = \frac{-0.3200}{0.1024} = -3.125$$

$$= \frac{-0.3 \times 0.2}{\sqrt{(-0.3)^2 \times (0.2)^2}} = \frac{-0.06}{0.09 \times 0.04} = \frac{-0.0600}{0.0036} = 16$$

→ Embedding ~~are~~ usually converted from -1 to 1.

$\boxed{\text{king - man + woman = Queen}}$

* Node in LlamaIndex is known as ~~the~~ chunk of documents. Node contains metadata of the document, their chunks and embedding



→ Indexing component in LlamaIndex executes chunking, Embeddings and storage.

→ Different Indexing.

i) Vector Store Index.
ii) Summary Index.
iii) Keyword Table Index.
iv) Document ~~by~~ Summary Index.

→ Vector Store Index is very popular and most common Indexing type.

→ Document Summary Index is very popular if we are working with multiple documents at a time. It enhances the performance of retrieval while working with multiple documents.

→ In this each documents are divided into chunks and then stored as node and then Docs summary and embedding.

→ We can customize the chunk elements like size, LLm models, embedding etc.

* Vector ~~Space~~ DB :-

vector DB stores vectors embeddings and other metadata for fast retrieval and similarity search.

→ Benefits :-

i) Data Management.
ii) Scalability.
iii) Real Time.
iv) Metadata Storage and Filtering.
v) Backup and Security.

Types of Vector DB.

i) Pinecone.

ii) Chroma

iii) Weaviate.

iv) Deeplake.

+ Tokenization is a process of breaking down para or article into smaller pieces (or chunks) to create a vocab.
+ LLM use subword tokenization.

Nikhil — 6 characters.

↓

Subword Tokens.

"Ni" - "kh" - "il" — 3 tokens

Prompt Engineering - II. (Part 2).

* **Chain of Density** :- It is mostly use for solving ~~the~~ text summarization problem. It ~~help~~ ensure that the generated summary are detailed yet concise and avoid information overload for clarity and comprehension.
* **Chain of Dictionary** :- This prompting technique generally use for machine translation task.
* **Chain of Symbol** :- This prompting technique ~~is~~ help in planning related task. It helps ~~is~~ user by replacing ~~plan~~ long description with easy symbols.
* **Chain of Explanation** :- This prompting is generally used for explaining hate speech.
* **Chain of knowledge** :- This prompting ~~helps~~ is generally used for knowledge augmentation
* **Chain of Emotion** :- This prompting is used for emotion emulation. Expression of player within the game.

Fine-Tuning :-

Fine tuning in GenAI is used to adapting pre-trained general-purpose model by training it further to specialize at given tasks by gaining domain knowledge. Align model to be helpful, harmless and honest.

→ Instruction finetuning :-

Training language models to follow instruction with human feedback.

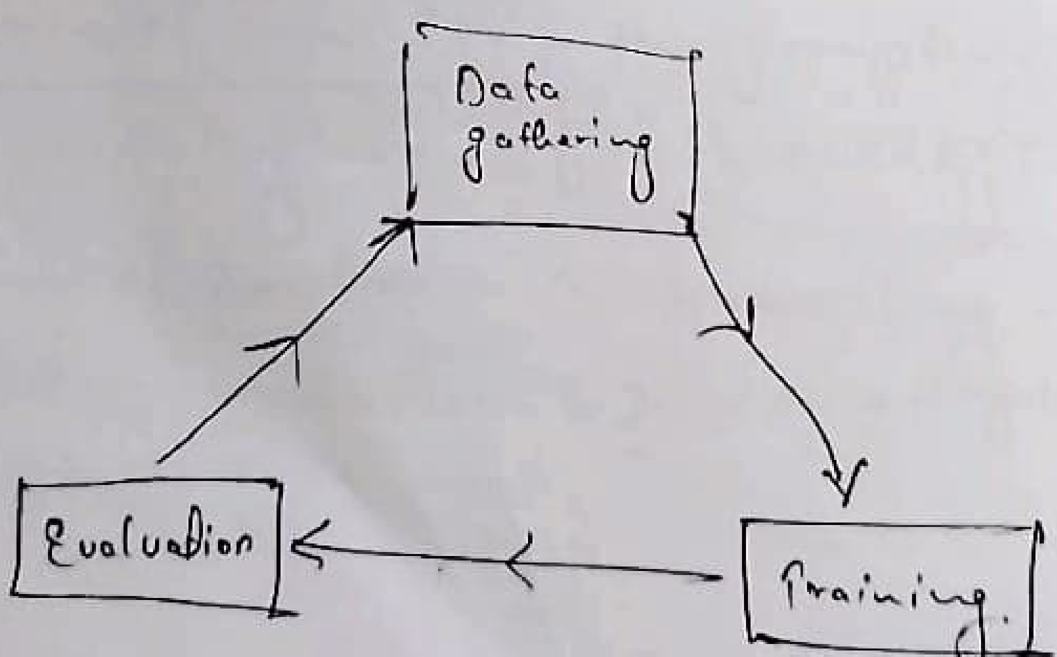
→ Training pre-trained LLMs to follow instructn

→ Process of fine tuning :-

→ Data gathering and preprocessing.

→ Training.

→ Evaluation.



- In Instruction fine-tuning dataset is created by skilled humans by annotating.
- Once the data is created then the base model is ~~fimed~~ fine tuned to learn from the instruction based dataset annotated by humans.

→ Another form of ~~dat~~ collecting a dataset is to use language model itself to generate data e.g GPT-4

→ One can use prompt template to generate dataset ~~tog~~ for fine tuning.

→ Another form is to use external sources.

* Best practices during fine tuning.

→ Quality.

→ Quantity (atleast 1000 of samples)

→ Diversity.

→ Source (human annotated is gold standard data).

* Data Collection Pipeline.

↳ Collect Instruction dataset

↳ Format and concatenate.

↳ Train/Test Split.

* Training Loop :-

Dataset → Forward
↓
Loss Computation
↓
Backpropogation
↓
Update weight.

→ Pytorch library can help in ~~training~~ fine-tuning model.
→ Trainer API and Accelerate from hugging face.

* PEFT :-

Parameter Efficient Fine Tuning.
→ It allow us to train large language model to perform well on specific task with fine tuning fewer parameters than full fine tuning.

→ PEFT approaches only fine-tune a small number of (extra) model parameters while freezing most parameters of the pre-trained LLMs.

→ It also reduce the risk of catastrophic forgetting. Catastrophic forgetting is a phenomenon where model tends to forget few knowledge post fine tuning.

→ It prevent overfitting

→

+ **Quantization:-**

Conversion from higher memory format to a lower memory format.

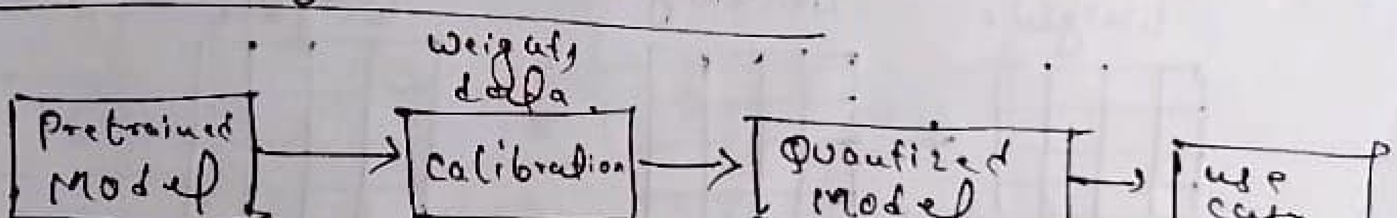
There are two types of quantization

i) Symmetric Quantization

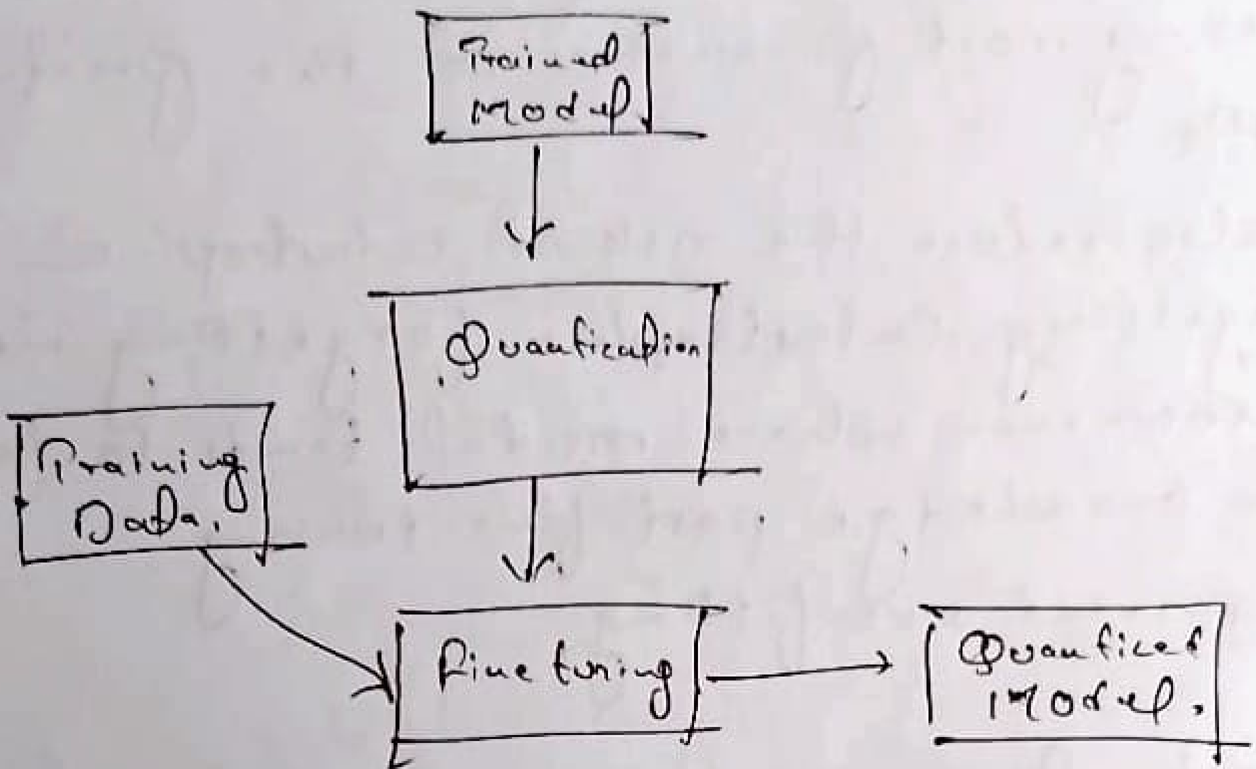
ii) Assymetric Quantization.

When we perform Quantization we depend on 2 things "Scale" and "Zero point".

+ **Post training Quantization:-**



Quantization Aware Training:-



* LoRA :-

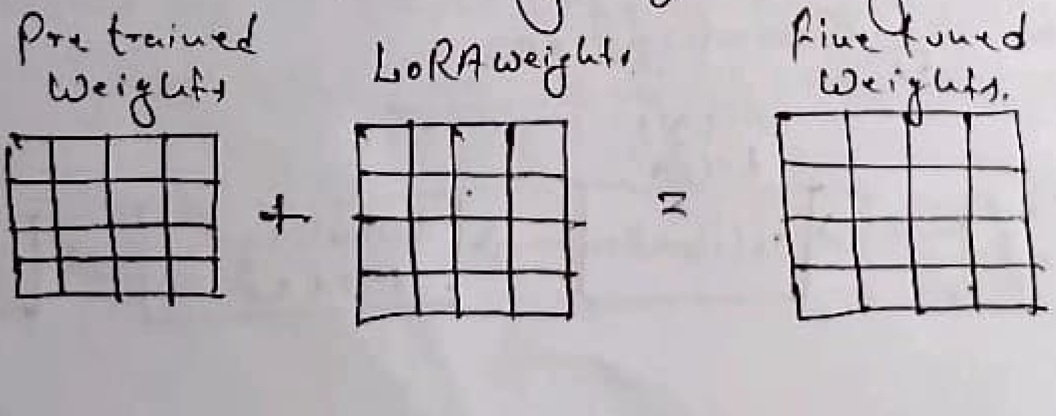
Low-Rank ~~Adaption~~ Adaptation of LLM

i) Full fine tuning.

ii) Domain Specific fine tuning.

iii) Task specific fine tuning.

In LoRA rather than ~~fin~~ updating all the weights we focus on updating few weights.



+ LoRA also update the weight of entire model but it perform matrix decomposition to save huge matrix into smaller one based on parameter called Rank.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

(3×3).

Pre trained Model weights.

$$B = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \qquad A = \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}$$

3 × 1 (1×3)

Rank = 1.

LoRA

$$\boxed{W_0 + \Delta W = W_0 + BA.}$$

W_0 = Pre-trained weights.

ΔW = ~~chain~~ changed weight.

Rank.	7B	13B	70B	180B.
1	167k	228k	529k	849k.
2	334k	456k	1M	2M.
8	1M	2M	4M	7M.
16	3M	4M	8M	14M.
512	86M	117M	270M	434M.

mostly used

If model wants to learn very complex things then it can use high rank.

* QLoRA is quantized LoRA by squeezing it to low precision.

* Quantization error is when we ~~quo~~ convert ~~a~~ high-precision model to low precision model. Then by de-quantizing the model again to original ~~a~~ model we see Quantization error. This is a loss that happens in the process. The less difference is the goal while quantization.

* <u>Model Training</u> :-

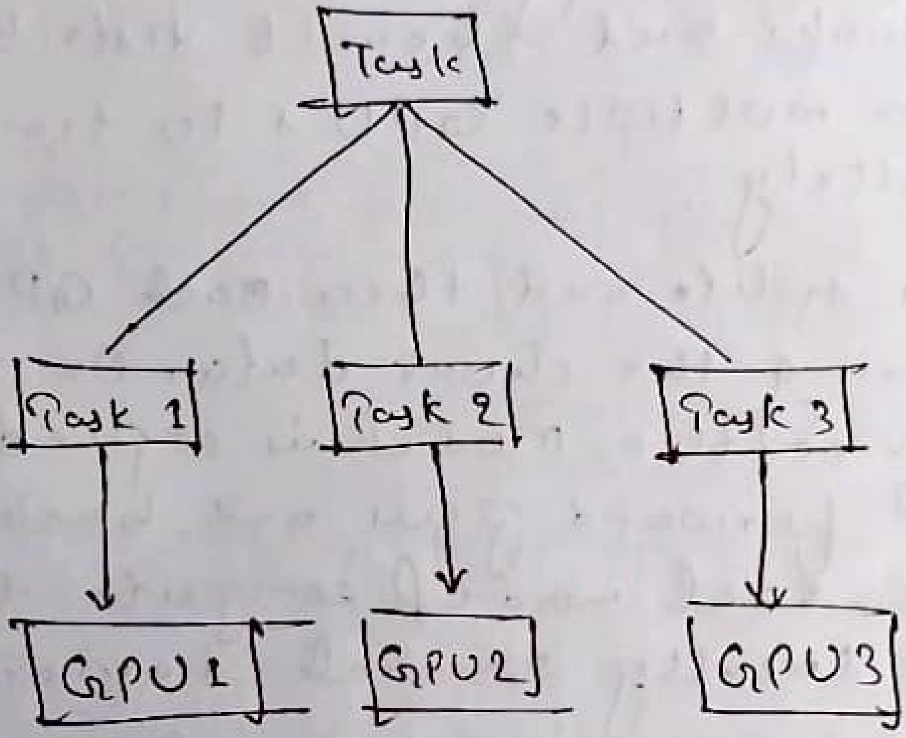
* Scaling ~~loss~~ laws is one of the core concept when we train our LLM from strach.

* Scaling laws define how much training data is required to train a LLM ~~for~~ of a particular model size. It gives you rough estimate of a training token for achieving the optimal model size

* Scaling laws ~~proper~~ most commonly known as chinchilla law.

* Parallel and Distributed Paradigm :-

In this process the ~~for~~ one task is divided into multiple sub task. Then each of these task run ~~on~~ parallely on seperate GPU's.



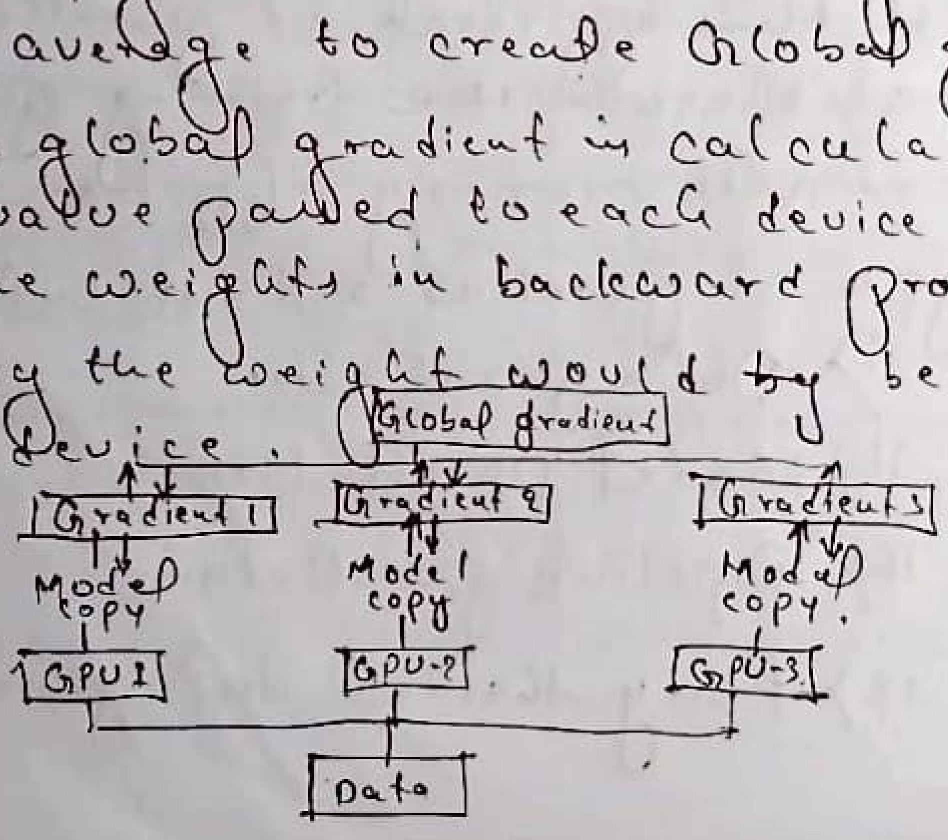
* With this approach of parallel and distributed it accelerate the training process faster.
* It is also memory efficient.
* Types of parallel and distributed technique.
  - i) Data parallelism
  - ii) Model parallelism.
  - iii) Pipeline parallelism
  - iv) Fully sharded data parallelism / ZeRO III

v) Tensor parallelism.

vi) 2D and 3D parallelism.

i) Data parallelism :-

- In this method training dataset split into multiple chunks and then it distribute the data on multiple GPU's to train the model parallely.
- The data is splits and then each GPU starts training the chunk data. On each GPU or device the model is copied in a manner of forward pass and backward pass where each model compute the gradient locally on each device.
- Then all local gradient combined together either by average to create Global gradient
- Then once global gradient is calculated then the value passed to each device to update the weights in backward propogation
- In this way the weight would be same for each Device.



ii) Model Parallelism :

→ With memory inefficient as limitation to ~~mod~~ data to parallelism. Model parallelism eliminate this limitation by splitting the model layer to multiple GPU's.

→ A model with eight layers segregated on GPU-0 and GPU-1.

| 0 → 1 → 2 → 3 | → | 4 → 5 → 6 → 7 → 8 |.

GPU 0 ↓ GPU 1

Communication Overhead

→ We know the model train in fou forward pass where each layer train and the pass the info to next layer. Once one GPU layer's are completed than it passes the info to another GPU using communication overhead

→ If all the GPU are on same machine then the communication overhead is faster or vice-versa.

→ Post loss calculation the backward pass ~~rotent~~ is done to updates the weights.

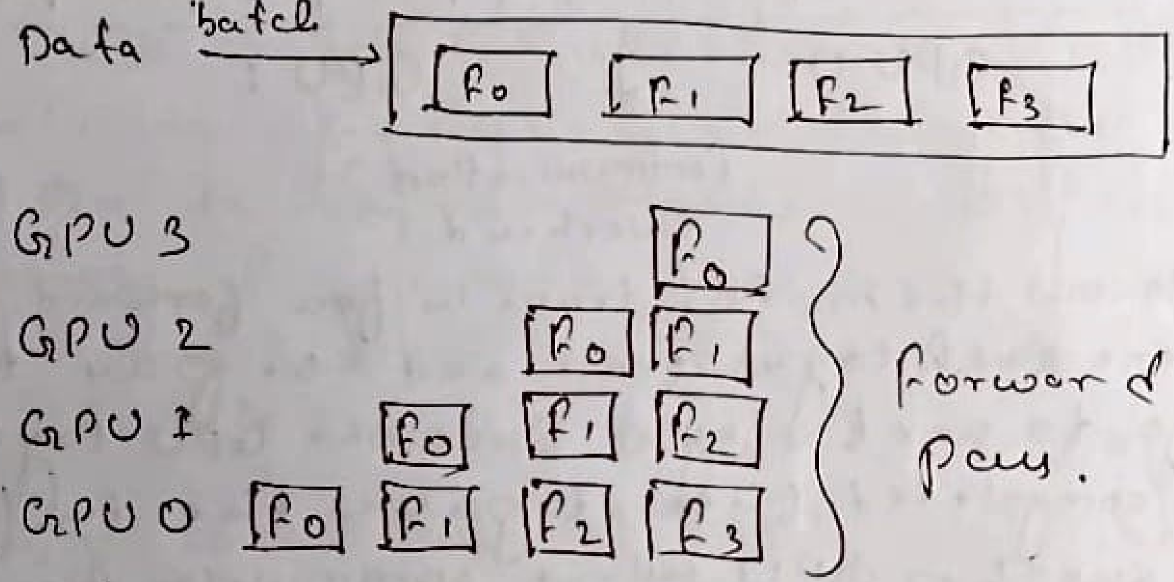
| 0 ← 1 ← 2 ← 3 | ← | 4 ← 5 ← 6 ← 7 | 8

~~GPO~~ GPU 0 GPU 1.

→ Limitation is that at a time only one GPU is active as there is dependency of forward pass.

iii) Pipeline Parallelism :-

- Pipeline parallelism is identical to model parallelism but it solves the GPU Ideal problem
- It create the incoming batch into microbatches and artificially create a ~~model~~ pipeline which allow different GPU to concurrently participate in the computation process.



iv) Fully Sharded Data Parallelism :-

The model parameter, optimizer are sharded across the GPU.

v) Tensor parallelism :-

In tensor parallelism the computations happening in the model are split across multiple GPU's.

$$\boxed{Y = GeLU(XA)}$$

vi) 2D and 3D parallelism :-

→ ~~2D and~~ In 2D parallelism we combine data parallelism along with ~~model~~ pipeline parallelism to leverage the advantage of both data and model sharding.

→ In 3D parallelism we combine pipeline parallelism ~~com~~ with tensor parallelism along with ZeRO parallelism.
e.g. Falcon, Llama uses 3D parallelism.

→ Steps involved in training LLM from scratch.

i) Training data curation
ii) Data preprocessing
iii) Tokenization.
iv) Model architecture.
v) Model Evaluation.

i) Training Data Curation :-

Process of collecting and aggregating the training data. The training data is a mixture of filtered web data and curated high ~~co~~ quality corpora.
e.g. stackoverflow, wikipedia, Github, Books etc.
Data collection is based on 3 principles:-
massive scale,
Diversity, High quality.

Common Crawl dataset is one of the source for LLM training dataset. Refined Web dataset from hugging face used to train Falcon

→ While collecting the data calculate the estimate training data size using scaling laws.

→ Focus on high quality of data.

ii) Data Preprocessing :-

In this step the focus is to create high quality data for better result.

common steps involved in data preprocessing

a) Data Sampling

Sampling dataset to handle the training distribution. Low quality data should be undersample and high quality data should be oversample.

b) Data deduplication

Removing duplicate ~~data~~ text from dataset.

[Jaccard Similarity method].

MinHash most common data deduplication ~~similarity~~ te method.

iii) Tokenization :-

This process is breaking down steps into sequence of tokens (numerical representation).

Tokenization methods.

a) word level

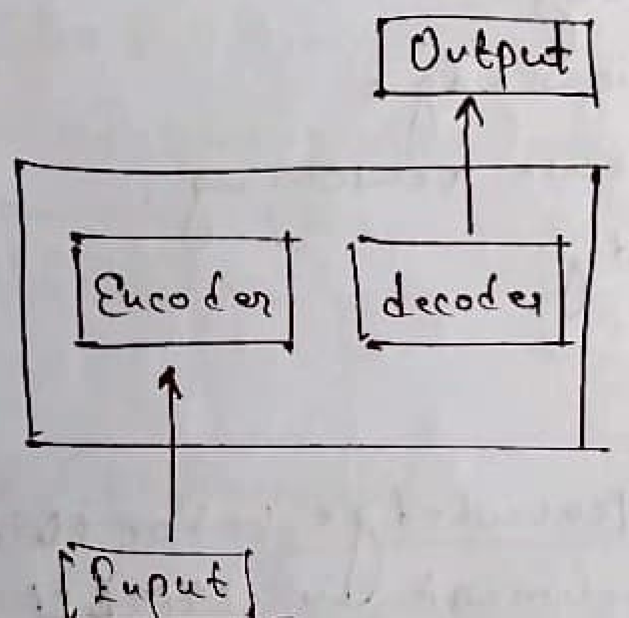
b) character level

c) sub-word level.

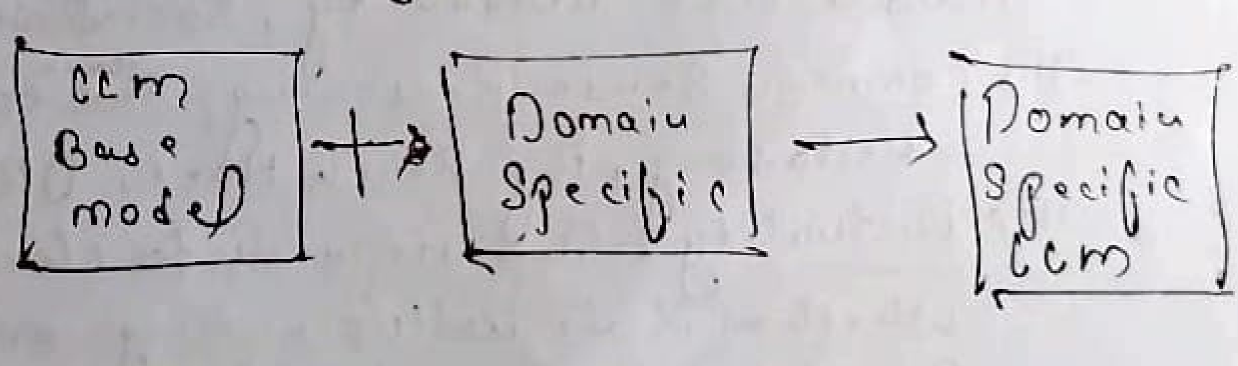
GPT-2 and 3 uses byte pair encoding algo for tokenization.

* iv) <u>Model architecture:</u>

Transformers are the backbone of LLMs



i) we can use existing model architecture



ii) Modify existing model architecture is achieved by modifying few layer and changing ~~hym~~ hyperparameters.

3.) Design new model architecture.

v) Model Evaluation:-

It is the process of measuring the performance of LLMs. ~~That~~ Model evaluation is based on lot of things:-

i) General knowledge
ii) Common Sense reasoning.
iii) Factuality
iv) Math
v) Code

i) General knowledge :- For this one of the famous benchmark is MMLU (Massive multitask language understanding). It consists of multiple MCQs from various sourses like humanity, social science, math.

ii) Common Sense Reasoning:- ~~two~~ three common bench Hellaswag, Big-Bench Hard, DROP.

iii) Factuality:- This steps is to check LLM whether it is hallucinating or not. One of the common benchmark is truthfulQA

iv) Math :- GSM8K and MATH benchmark.

School math 8K. Algebra Geometry.

v) code :- HumanEval and Natural2Code.

There are 3 methods to evaluate LLM on these benchmarking.

i) Zero Shot prompting :- Provide no hint to the model to answer the question.

ii) Few Shot prompting :- With few examples.

iii) Chain of thought prompting :- provide problem and then detailed step-by-step to arrive at answer.

* LlamaIndex Continuation.....

→ Node parsers :- It convert raw data in structural node suitable for structuring and retreival

→ Node parser break down data into manageable prices(node) based on specific rules or formats such as HTML, JSON or CSV.

→ Types of node parsers.

i) Text Node Parser

ii) HTML Node Parser.

iii) JSON Node Parser.

iv) CSV Node parser.
v) Markdow Node Parser.
vi) Hierarchical Node Parser.
vii) SimplefileNode parser.
viii) SimpleNodeParser.

* Vector_Store save the embeddings. however docstore save the actual metadata of the file.

* Default chunk size is 1024 which is customized.

* Vector DB Search technique:-

i) Locality-Sensitive Hashing (LSH).
It group similar item into same bucket based on hashing function. Similar items are grouped together by ~~hash~~ (fig)

ii) HNSW :- It uses graph to reach to actual point.

iii) Annoy.

* Evaluation of Metrics:-

While working with RAG System we consider lot of factors like chunk size, LLM, Embedding model, Retrieval Algorithm, Response Synthes.

* Two most important evaluation metrics are Retriever and Response Synthesis.
* The performance of the Retriever is evaluated by Hit Rate and Mean Reciprocal Rank (MRR)
* For Response Synthesis we can evaluate faithfulness, Correctness, Context & Answer Relevancy, Symantic Similarity.

* Hit Rate :-

The fraction of queries where the correct context is found within the top-k retrieved contexts.

Query	Actual Node	Retrieved Nodes	Hit Rate
Q_1	N_1	N_2, N_3, N_1	1
Q_2	N_3	N_2, N_3, N_5	1
Q_3	N_3	$\underbrace{N_1, N_2, N_4}_{\text{Order.}}$	0

Final Hit Rate = (1+1+0)/3 = 66.7 %.

* MRR :-

Takes into account the position of the actual node in the retrieved set of nodes. Evaluates the system's accuracy by looking at the rank of the highest-placed relevant nodes.

Query	Actual Node	Retrieved Node	Order	Reciprocal R..
Q_1	N1	N2, N3, $\boxed{N1}$	3	1/3
Q_2	N3	N3, N5, N7	1	1/1
Q_3	N3	N1, N2, N4	0	0

$$MRR = \left(\tfrac{1}{3} + \tfrac{1}{1} + 0\right) / 3$$

$$= 4/9 = 44.4\%$$

$$\text{Hit Rate} = (1 + 1 + 0) / 3 = 66.7\%$$

* **Faithfulness :-**

The faithfulness evaluates whether the generated response is in with context or some kind of hallucination. It can be done by using faithfulness score

$$\text{faithfulness} = \frac{\text{Number of claims in response with context}}{\text{Total number of claims in generated answer}}$$

Score ranges from (0, 1). This can be done by evaluating each statement.

Statement 1 : Yes

Statement 2 : No.

$$\text{Faithfulness} = 1/2 = 0.5$$

* **Relevancy:-**

It checks how relevant the generated response is with the given query.

* **Correctness:-**

Response generated VS Actual response.

* **Different Indexing:-**

* **To overcome the limitation of RAG System:-**

i> **Data Augmentation:-** Enrich our dataset by storing supplementary information beyond raw text chunks, such as metadata or stru. data

ii> **Multipurpose use of LLM:-** We can use LLM for task beyond text generation.

iii> **Advanced Retrieval Technique:-** Go beyond basic top-k embedding lookup and implement advanced retrieval methods.

iv> **Optimized Embeddings:-**

* Advanced Retriever technique:-

i) Sentence Window Retriever:-

→ It extract relevant sentences or short text from unstructured text data.

→ It uses a concept of sliding window to extract overlapping text.

→ Compute embedding for each segment and find most similar segment to query.

LlamaIndex doesn't have any inbuilt Sentence Window Retriever.

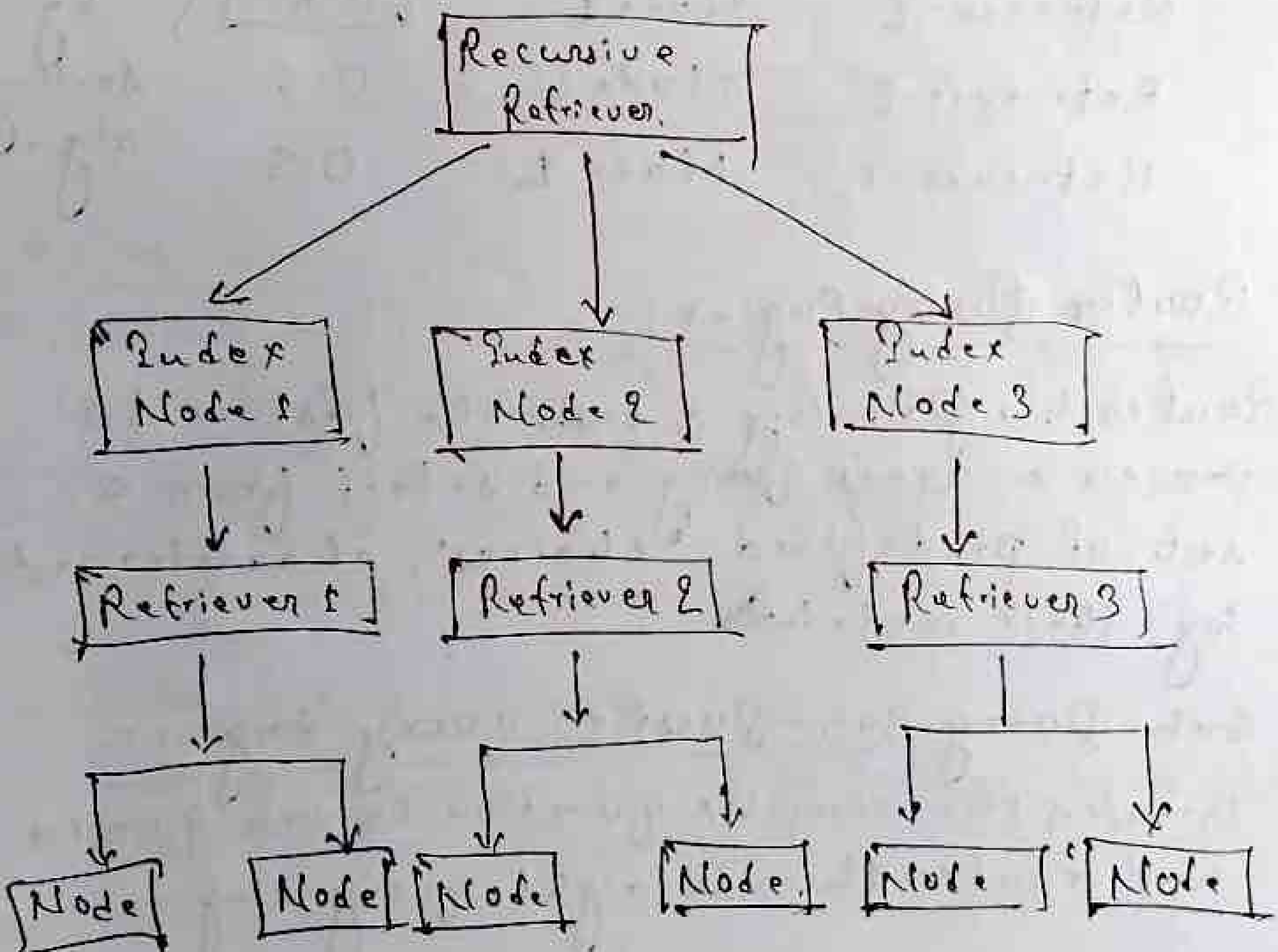
Sentence window Retriever = SentenceWindow NodeParser + MetadataReplacementNode PostProcessor.

ii) Automerging Retrieval :- For the better result of retrieval, automerging retrieval use Hierarchy Node parser to first define the nodes ~~on~~ in ~~parent node~~, Root Node, Parent Node and then Leaf Node.
It ~~m~~ automerges ~~parent~~ leaf Node to create a ~~root~~ parent node to cover more context and better retrieval. It helps our retrieval to give enhanced context.

iii) Auto Retriever:-

The idea behind auto retriever is rather than getting top-k nodes ~~we can~~ based on the similarity search using vector Index. We can also retrieve the top chunks based on the metadata filters. Now these metadata filters retrived using large language model and these chunks used to have metadata info. It can be implemended using chromadb, Pinecone

iv) Recursive Retriever:-



* **Hybrid Fusion Retriever:-**

→ In this multiple Retriever Outputs are combined together to create one powerful retriever.

- Each retriever select top 2 nodes based on the similarity score.

→ Once done then sorting algorithm select top k nodes based on the ~~retri~~ similarity score.

Retriever:	Node	Score	
Retriever-1	Node 1	0.9	Selected by sorting algo
Retriever-2	Node 1	0.8	
Retriever-2	Node 2	0.6	
Retriever-1	Node 2	0.5	

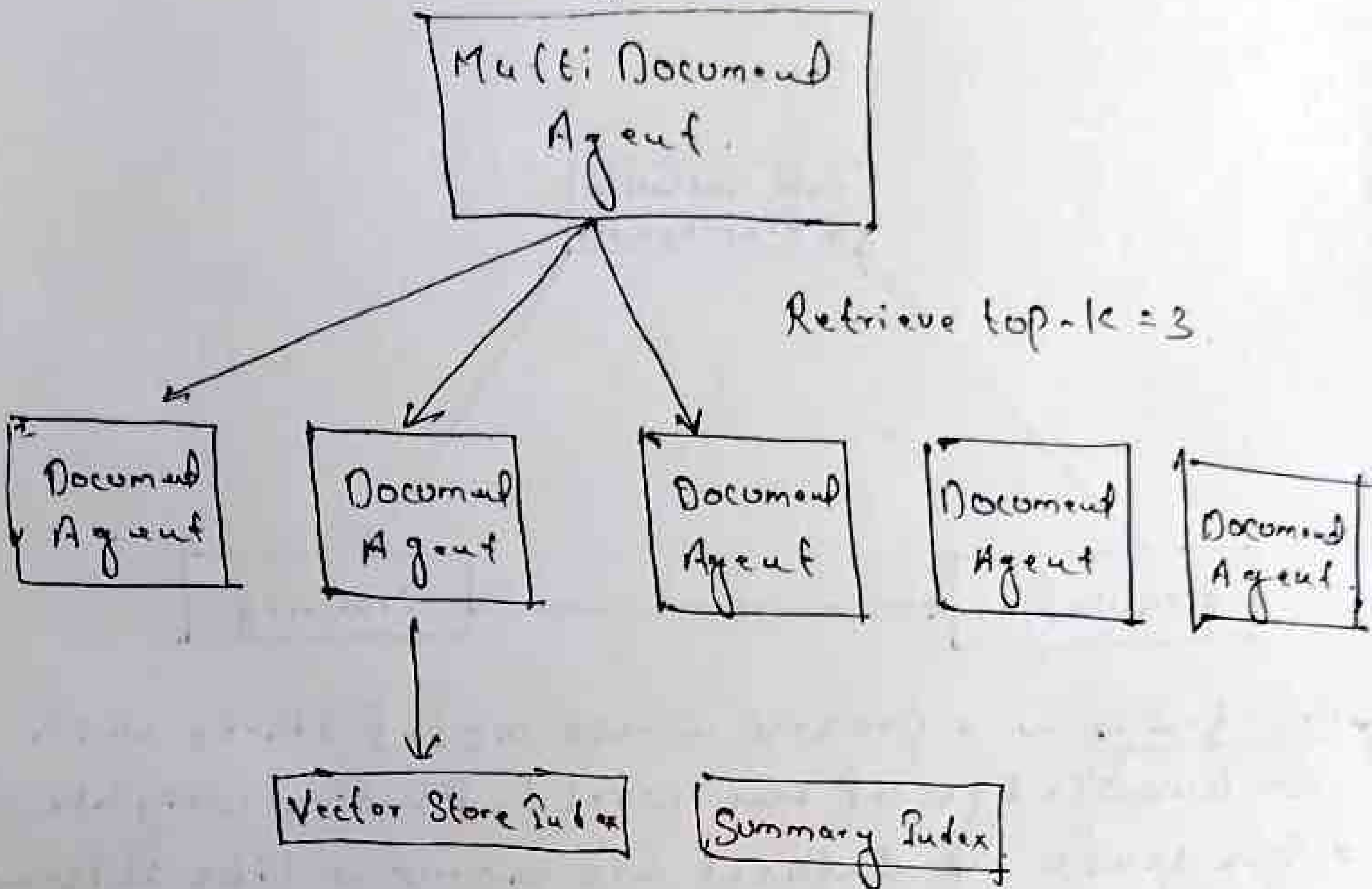
* **Router Query Engine!**

Router has primilary 2 tasks the first is to process a user's query and select from a set of predefined "choices", characterized by their metadata.

* ~~Sub-Query~~ **Sub-Question query Engine:-**

Breaks the complex question to sub queries and routes them to right query engine.

* Multi Document Agent :-



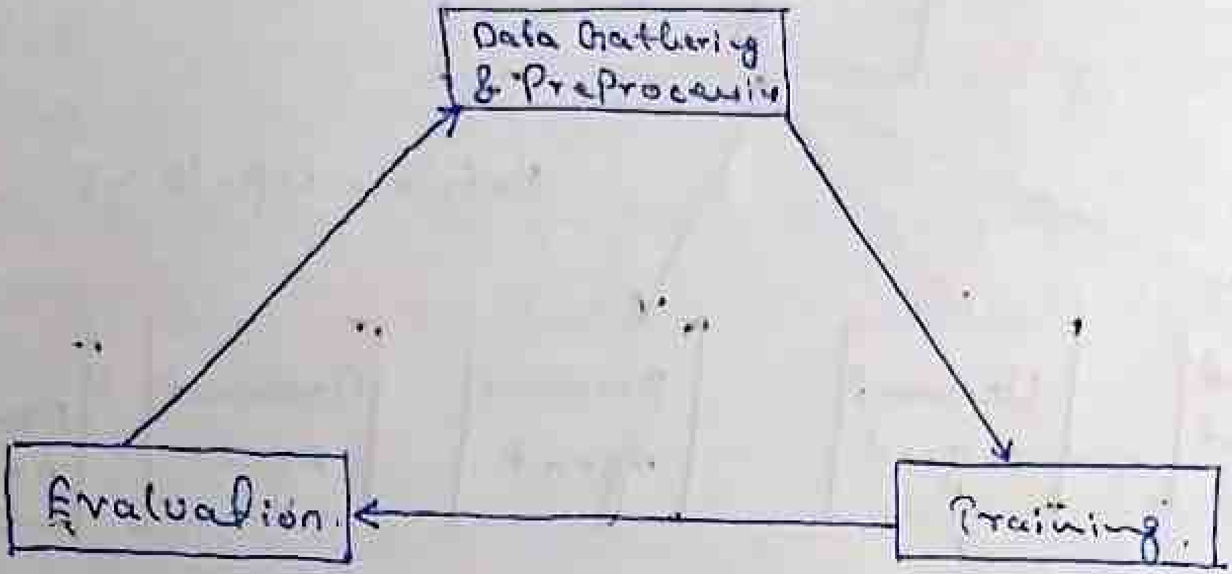
* Fine Tuning LLM.

Adapting pre-trained general purpose mode by training it further to :-
- Specialize at given task(s) by gaining domain knowledge.
- Align model to be helpful, harmless and honest.

* Instructing fine tuning :-

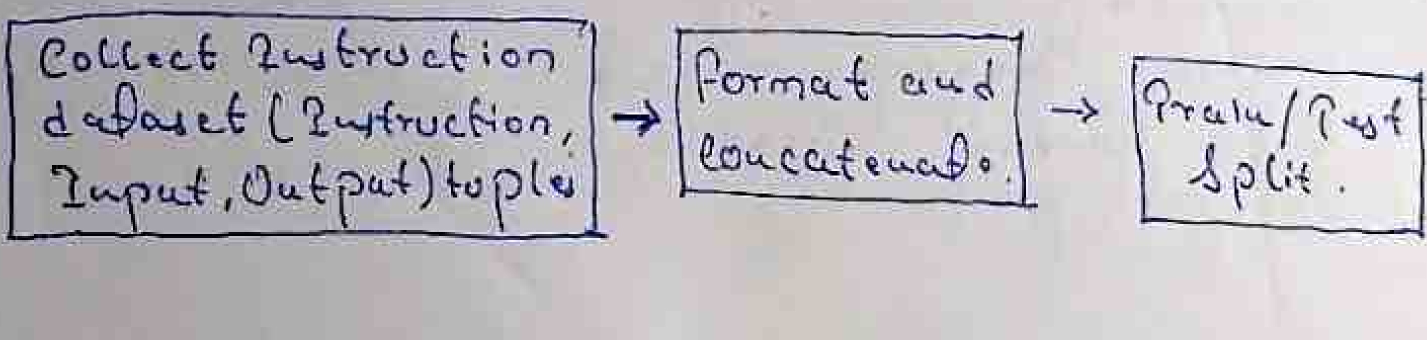
Training language models to follow instructions with human feedback.

Finetuning Process.



- **Pre-training** is a process where model starts with no knowledge of the world. Random weights.
- The scale of dataset are enormous like Internet data (30 Trillion tokens and 20B documents).
- **Finetuning** model starts with pre-trained model which has general world knowledge and language skills.
- Required much less data. The numbers of sample are in the order of thousands.

* **Data Collection Pipeline.**



Prompt Engineering :-

i> Persona :- This is a case where we ask LLM to act as a specific person like physicist, mathematician etc.

ii> Context :- The context provides the necessary background information to give more accurate.

iii> Examples :- In this we provide example or sample like how Input would look and output would look like.

iv> Instruction :- A specific type like I need in this format, 2 lines, in bullet points etc.

v> Format :- We can ask for different format like Json, html, csv, table etc.

vi> Constraints :- We can apply any limitation like limit points, limit character count etc. ~~limit~~

vii> Tone :- Style in which output can be generated like formal tone, polite tone, don't use words like kick-off, dear.

viii> Delimiters :- special symbol or character which segregate Input, ~~or~~ example, code etc.

→ LeChat from Mistral AI.

- Flipped Interaction Pattern:- In this gpt ask question to get started for specific task.

- Directional Stimulus Prompting:- Where we ask GPT to ~~a~~ focus on follow ~~the~~ topic when generating text or summarization.

- Semantic Filter pattern:-

* Stable Diffusion Model:-

→ Stable diffusion model is used to generate Image ~~co~~ from text.

→ It is state of art model.

→ Inspired by physics of fluids / gases.

→ Stable diffusion doesn't convert text to image it converts noise to image. By the guidance of text by performing reverse diffusion.

→ It understands both text and Image.

→ When any image gets generated from stable diffusion model and then ~~we~~ when we want to do slight modification using text like change the ear of the person. This change is called reverse diffusion.

* Reverse Diffusion:-

→ Diffusion in image is a diffused states ~~too~~ that can be imagined as - True noisy image.

→ It converts the noisy image to denoisy image until it is not making sense as per the text entered by user.

Noisy Image. Bird: after many reverse diffu.

Noisy Image → Neural Network. → Less Noisy Image.

At every step of Reverse diffusion

→ Calculate the noise on current Image.

→ Remove the noise from current Image.

→ Repeat.

$$\boxed{C \Rightarrow R \Rightarrow R}$$

Calculate Remove Repeat.

* Diffusion is slow and it takes long time which would eventually required more compute resourse to speed up the process it uses latent. In this we take the image and convert to vector image called latent and the we perform denoising on lantent and then finally we convert it to Image again.

CLIP :- With the help of CLIP functionality Stable diffusion model ~~ero~~ create image from text.

- → Contrastive Language Image pre-training trains an Image encoder and a text encoder to predict correct pairing.
- → CLIP is trained encoders. It converts the ~~i/p~~ input from user into encoded vector and then image also to encoded vector. post that it computes the dot product of it and then take the sum of highest dot product value to generate image.
- → The dot product is cosine similarity.
- → Contrastive loss :- When ~~we create~~ the encode text vector and encoded image vector is ~~mathin~~ matching then the contrastive loss value would be less and vice-versa.
- → ~~De~~ Encoding and decoding of image is taken care by Variational Autoencoder (VAE).

* While training stable diffusion model we ~~creat~~ add noise to the image and create multiple version of it and then it passed to model to training. The noised Image sent to U-net and then U-net tries to predict the amount of noise in it.

→ as we have added the noise ~~then~~ so we compare the added noise with the predicted noise from U-net. Calculate the loss and based on feedback it keeps on updating.

→ This process is DDPM - Denoising Diffusion Probabilistic Model.

→

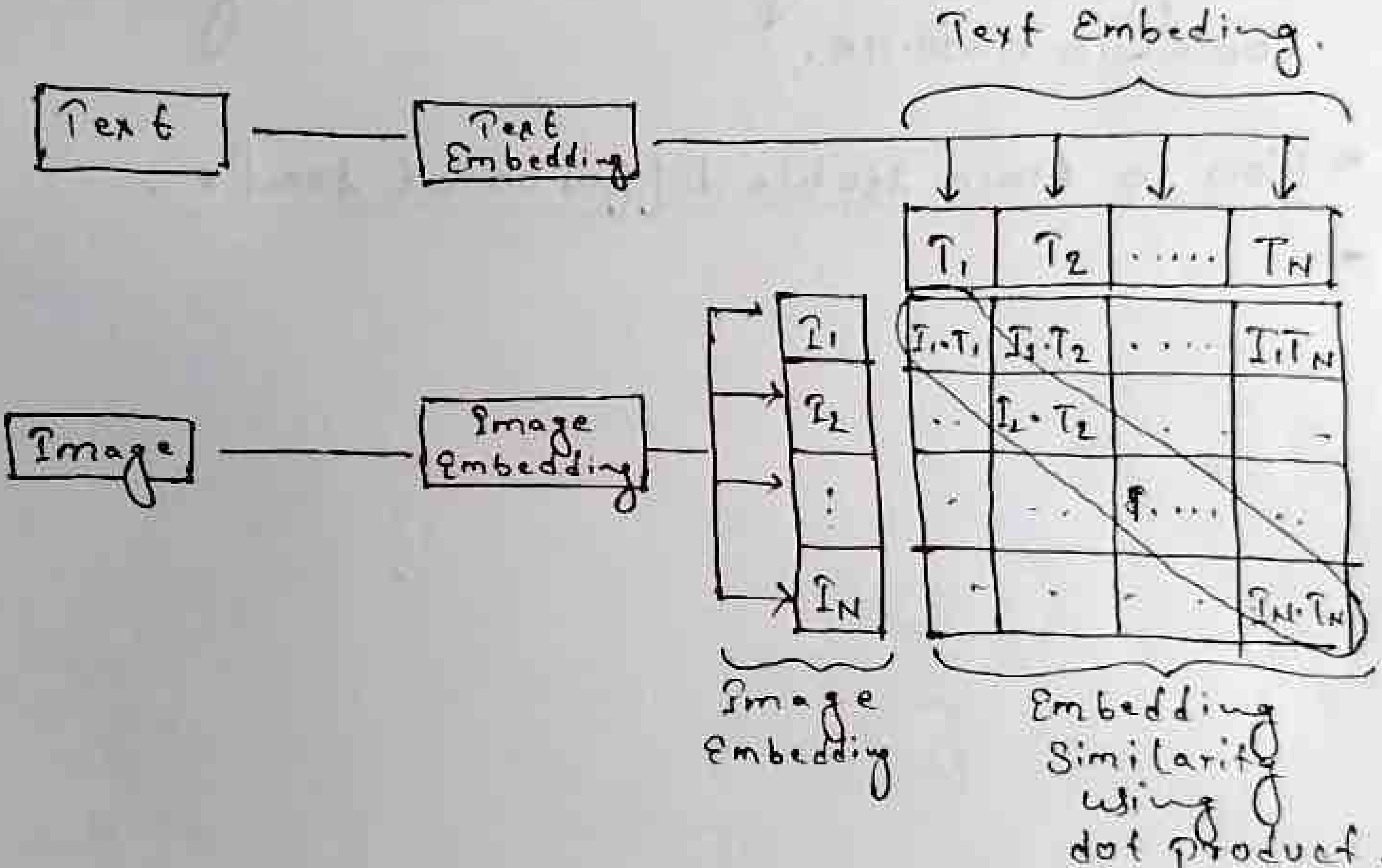
* VAE (Variational Autoencoder)

→ VAE is ~~used~~ commonly used in image processing, image enhancement etc. applicati It is also used in stable diffusion model.

→ common application of VAE is Denoising, In-painting.

* In stable diffusion model when user enters a text then the text converted to text embedding and then with the help of prior it converts the text embedding to image embedding. Post that with the help of Decoder the image embedding converted to target image.

→ CLIP has two encoders text encoder and image encoder.



* "Prior" and "Decoder" both are diffusion based model.

* Text prompt converts to Text Embedding via trained text encoder.

* A prior model maps the text embedding to a corresponding image.

* An Image decoder stochastically generates image.

* **Glide :-**

Decoder is a diffusion model called GLIDE by openAI. With the help of GLIDE model the generated photos was very realistic look in nature.

* **How to train stable diffusion at scale :-**

→