Large Language Models

Dr Shantanu Pathak

What are Large Language Models (LLMs)?

- - LLMs are advanced AI systems trained to understand and generate human-like text.
- - Examples: GPT (OpenAI), BERT (Google), LLaMA (Meta).
- - Key Feature: Ability to process and generate natural language at scale.

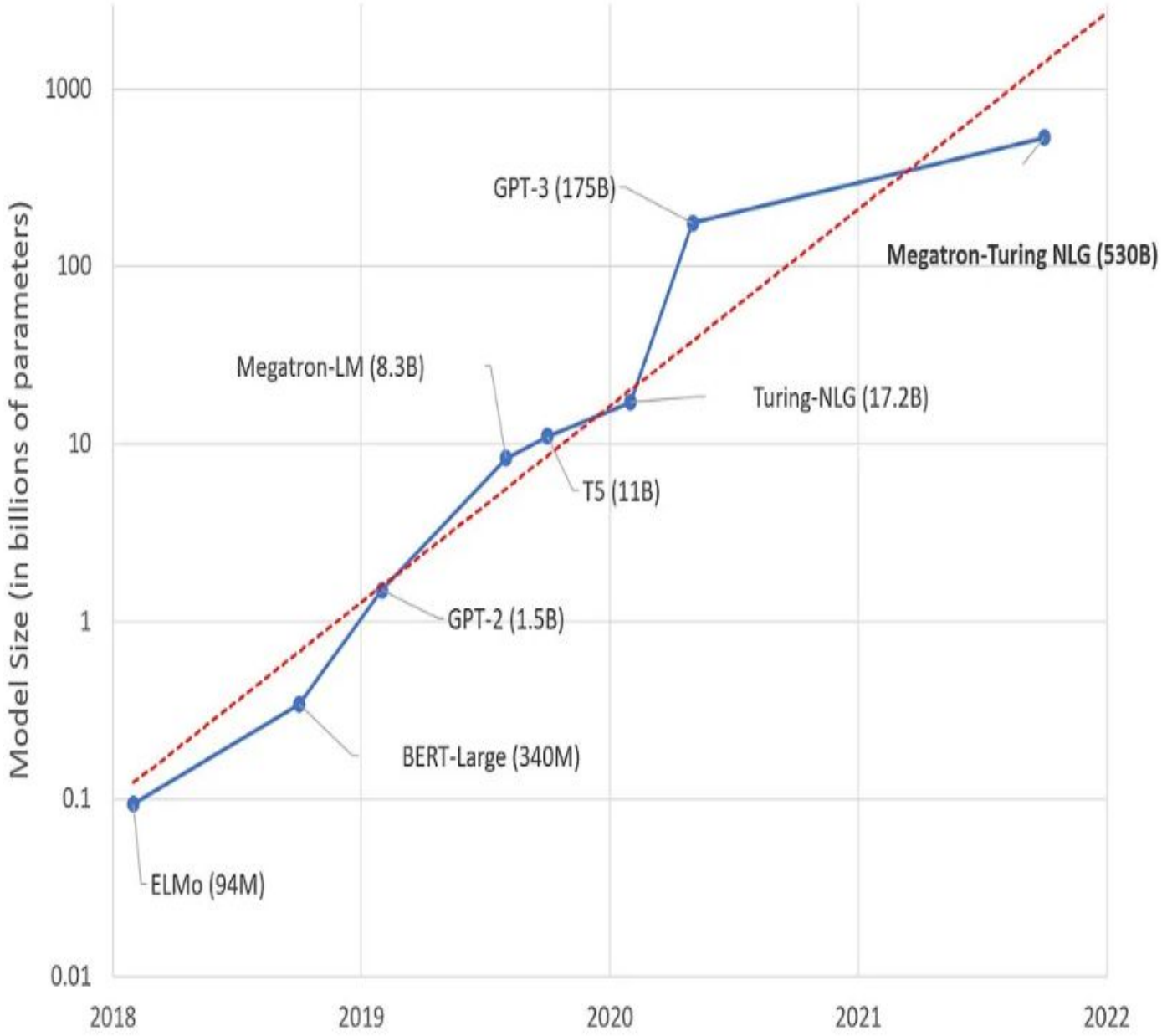
LLM

- A *language model* is built to process and understand a text input *(prompt)*, and then generate a text output (response) accordingly. These models are usually trained on an extensive corpus of unlabeled text, allowing them to learn general linguistic patterns and acquire a wide knowledge base. The primary distinction between a regular language model and a *large* language model lies in the number of parameters used.

- LLM should have min 10 billion parameters (total weights and bias)

How Do They Work?

- - Input-Output System: Takes text input and produces text output.
- - Training Data: Massive datasets from books, websites, and other sources.
- Based on Transformers
- Growing number of parameters of LLM



Applications of LLMs

- - Industries:
  - • Healthcare (medical chatbots)
  - • Education (tutoring and content generation)
  - • Business (customer support, content marketing)
- - Functions:
  - • Text summarization
  - • Language translation
  - • Sentiment analysis
  - • Code generation

Benefits of LLMs

- - Efficiency: Automates repetitive text-related tasks.
- - Scalability: Handles large volumes of data seamlessly.
- - Accessibility: Bridges language barriers and improves accessibility.

Challenges

- - Ethical Concerns: Potential for bias and misuse.
- - Resource Intensive: High computational and energy demands.
- - Accuracy: May generate incorrect or nonsensical outputs.
- - Data Privacy: Concerns with data used in training.

Future of Large Language Models

- - Advancements:
  - More efficient architectures.
  - Improved fine-tuning for specialized tasks.
- - Ethical AI Development:
  - Reducing bias and enhancing transparency.
- - Integration with Other Technologies:
  - AI systems like robotics and AR/VR.

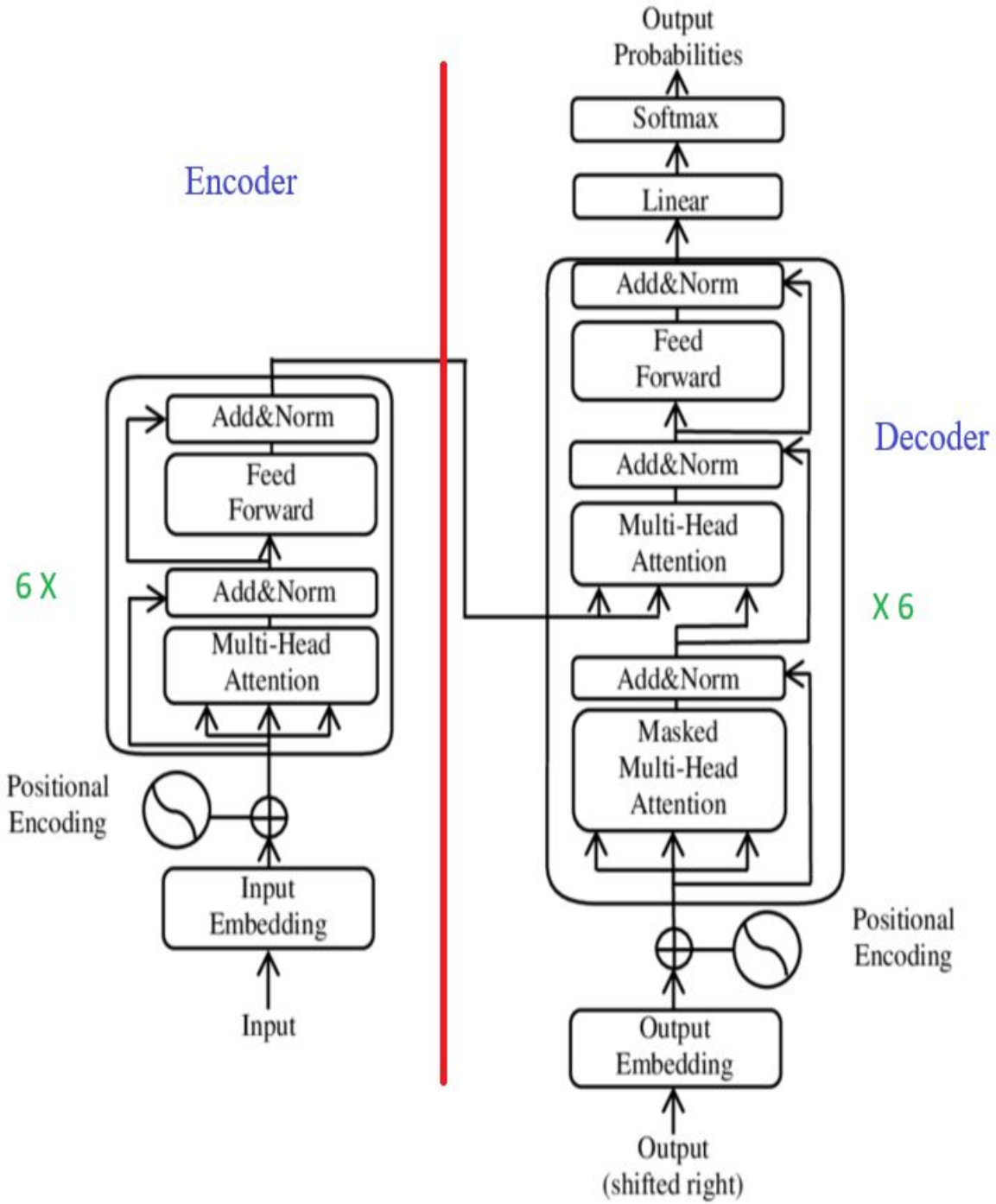
History of LLM

- Tf-idf
- Word Embedding
- RNN for classification of NLP
- RNN seq2seq models for summary generation, translation tasks
- seq2seq with self attention
  - simple self attention
  - scaled dot-product self attention
- Transformer Architecture
- Encoder ONLY models (BERT ..etc)
- Decoder ONLY models (GPT, T5, LLaMa, Gemini ..etc)

LLM Technical details

Transformer Architecture

- Tokenization algorithm
- Positional Encoding
- Multi-head Self attention
- Layer normalization
- Feed Forward Layer
- Encoder-Decoder Working
  - 6 encoders and 6 decoders
  - Working in parallel
- Output Layer (SoftMax)



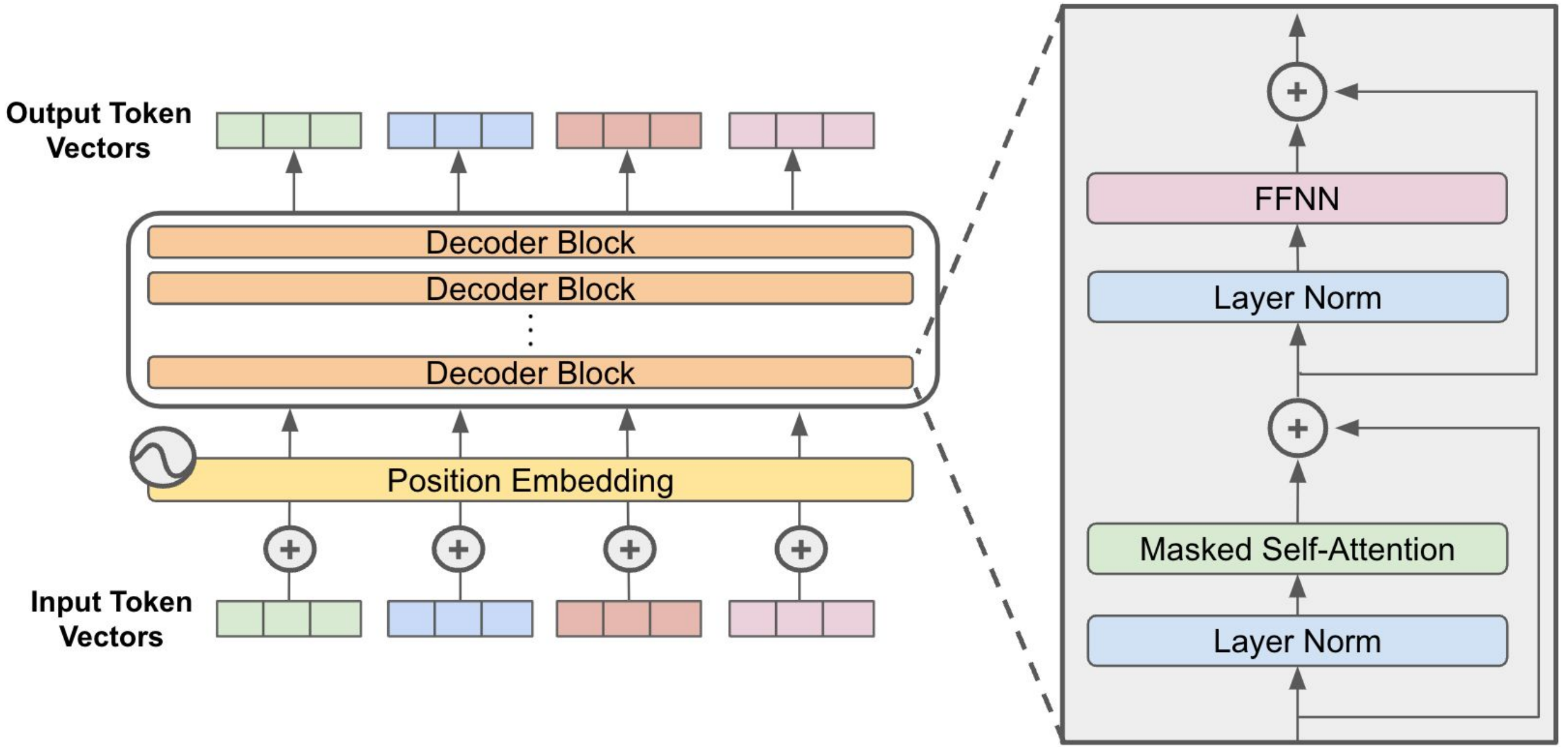
Encoder ONLY models (BERT ..etc)

- BERT: **Bidirectional Encoder Representations from Transformers**
  - masked token prediction and next sentence prediction
- Only Encoders are used
- Encoders extract features from the given text
- Extracted features can be used to predict fill in the blank (masked word) or next word or next sentence

Decoder ONLY models

- GPT (Generative Pre-trained Transformers)
  - Uses decoders only
  - Every decoder has following layers
  - normalize (layer normalization)
  - multi-head attention layer
    - "Looks at" and combines information *across* all positions
  - Add [used for skip connection]
  - normalize (layer normalization)
  - Position wise Feed Forward Neural Network
    - "Processes" information *within* each position independently
  - Add [used for skip connection]

  - GPT-2 has 48 decoder layers and GPT-3.5 has 96 decoder layers


Output Token Vectors
Decoder Block
Decoder Block
Decoder Block
Position Embedding
Input Token Vectors
FFNN
Layer Norm
Masked Self-Attention
Layer Norm

Decoder :: Multi-head attention layer

- Self-Attention
  - Calculate how one word is related to all other words in given text
  - One head is used to have one way to find the relations
  - Multiple heads allows extracting multiple perspectives for same word in same text
- Self-Attention Types:
  - Simple Self Attention
  - Self-Attention (Scaled Dot Product Self Attention)
  - Multi-head Self Attention
  - Masked Self-Attention
- Self Attention Operation
  - Refer the diagram on the board
- Multi-head Self attention
  - Output Z of every head is concatenated and multiplied with Wo matrix to get output matrix same size as Word embedding matrix
- Masked Self-Attention
  - Used in decoder to hide future input words so that decoder can learn to predict next word
  - Also used in BERT decoder to mask specific words so that BERT learns to predict those words like fill in the blanks
- GPT-2 uses 12 heads, GPT-3 uses 96 Attention Heads

Decoder :: Position wise Feed Forward Neural Network

- For each token (word) in input we have one feed forwards NN associated
- FFNN :
  - Input Layer (Same as size of word embedding d)
  - Dense layer with GeLU Activation (4 * d no of neurons)
  - Dense Layer with Linear Activation (d neurons)
- So, FFNN creates new representation for every token (word) separately
- FFNN introduces non-linearity in output to solve complex problems
- Finally output size is same as word embedding matrix

Decoder :: Add layer and Layer Normalization

- Layer Normalization
  - It is used to normalize inputs before multi-head self attention or FFNN
- Add Residue Connection
  - Original input values are added to the output of multi-head self attention or FFNN
  - Residue connection is used to avoid vanishing gradient problem, information preservation and make the gradients stable
  - This is elementwise addition of normalized matrix and original input matrix
  - Out size remains same as the input size

LLM Output Generation (Linear + Softmax Layer)

- After all decoders are applied, final decoder generates a ‘z’ matrix
- This matrix is passed to dense layer with linear activation
- This dense layer has no of neurons same as vocabulary size
- This linear layer converts shape of ‘z’ (seq_len, d) to (seq_len, vocabulary_size)
- Output is considered as probabilities of next tokens for each word.
- We are interested ONLY in probabilities at the last
- These probabilities at the last are given to Softmax activation to normalize
- Final output is normalized probabilities of all words in vocabulary
- Now we select the next token

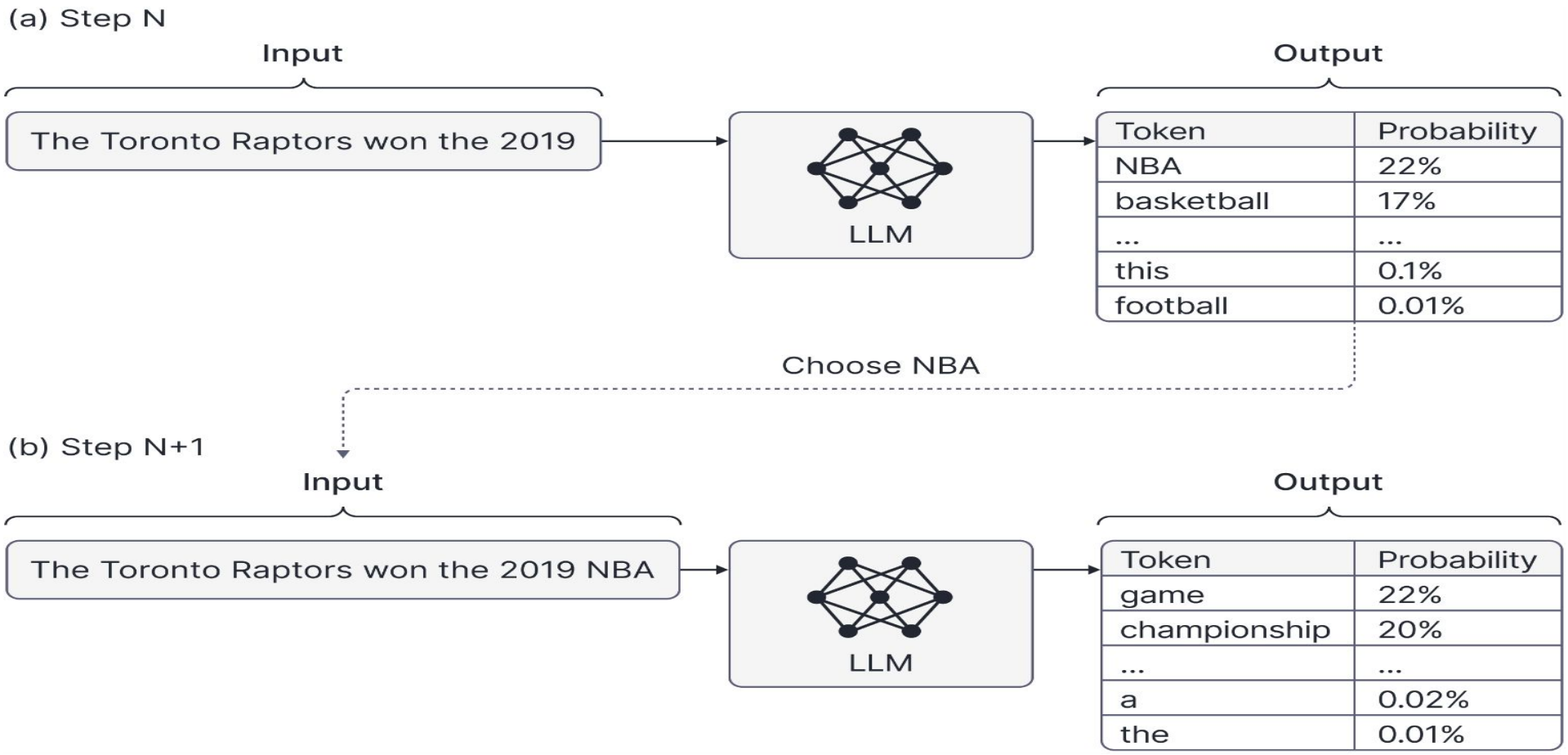
LLM Next word selection using probabilities

- (i) selecting the most likely token
  - Limitation: creates general outputs and difficult customize as per user input
- (ii) randomly drawing a token according to the probabilities.
  - Limitation: While writing longer sequences, words may be selected out of current context. So writing may look like dis-continued words / sentences
- (iii)Select top-k tokens based on probability and some rank / threshold
- (iv)Temperature scaling (generally between 0 to 1)
  - Low temperatures render outputs that are predictable and repetitive. Conversely, high temperatures encourage LLMs to produce more random, creative responses.
  - Temperature controls softmax operation on predicted probabilities of the tokens. Low temperature will just normalize probabilities as in general. But, higher temperature all probabilities will be converted to almost same value.
  - Formula for temperature :
  - $qi = \exp(zi/T) / \sum \exp(zj/T)$
- (v)Beam-Search::
  - Select the token based on calculation of ‘n’ next tokens for every token

LLM Steps in summary

- Most LLMs are *auto-regressive* language models
- This process involves following key technical components.
  - 1. input *token embeddings and positional encodings*
  - 2. Find the relevance and context of input using self attention (scaled dot product self attention, trained weights and bias, external sources, fine tuning)
  - 3. probability of the next word predicted based on the preceding text provided as the input
  - 4. Next word selection process
  - 5. repeat step3 and 4 till relevant output is completely generated OR output token limit reached
- This allows them to generate extensive bodies of text

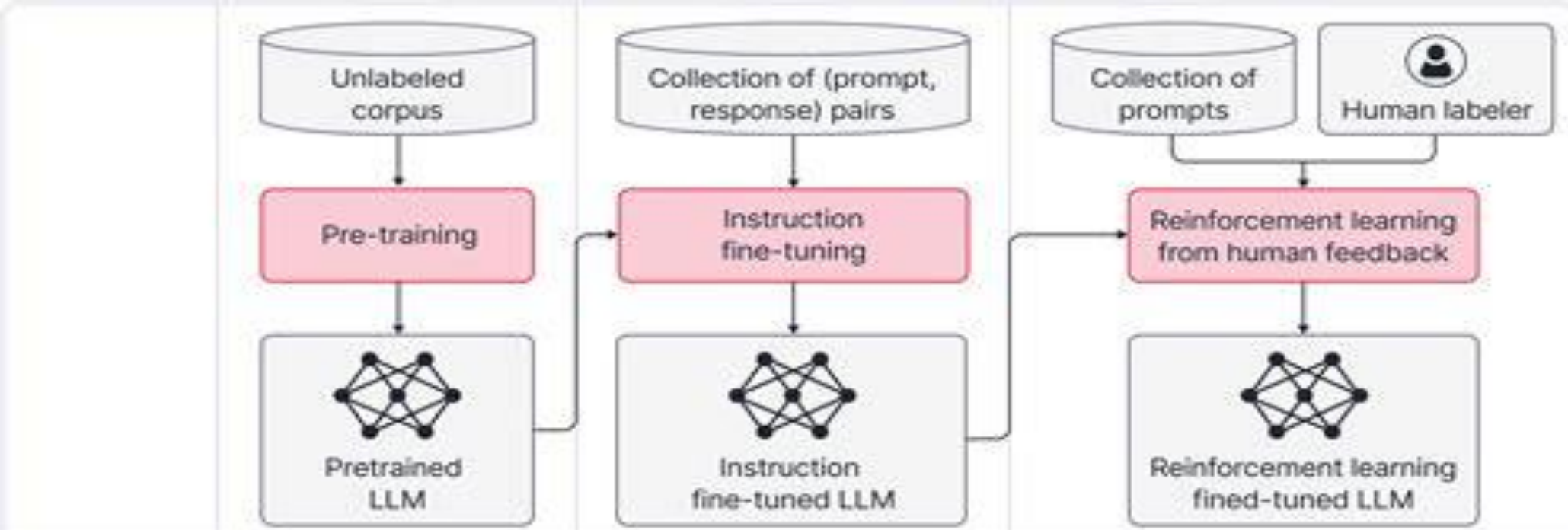
LLM Steps


(a) Step N
Input
Output
The Toronto Raptors won the 2019
LLM
Token
Probability
NBA
22%
basketball
17%
...
...
this
0.1%
football
0.01%
Choose NBA
(b) Step N+1
Input
Output
The Toronto Raptors won the 2019 NBA
LLM
Token
Probability
game
22%
championship
20%
...
...
a
0.02%
the
0.01%

LLM Training as a chat-bot

- Using Instructor based Fine tuning / Prompt Engineering
- Reinforcement Based Fine tuning (RLHF, RLAIF)
- Direct Preference Optimization (DPO)

LLM Prompt based fine tuning (3 Steps of training)

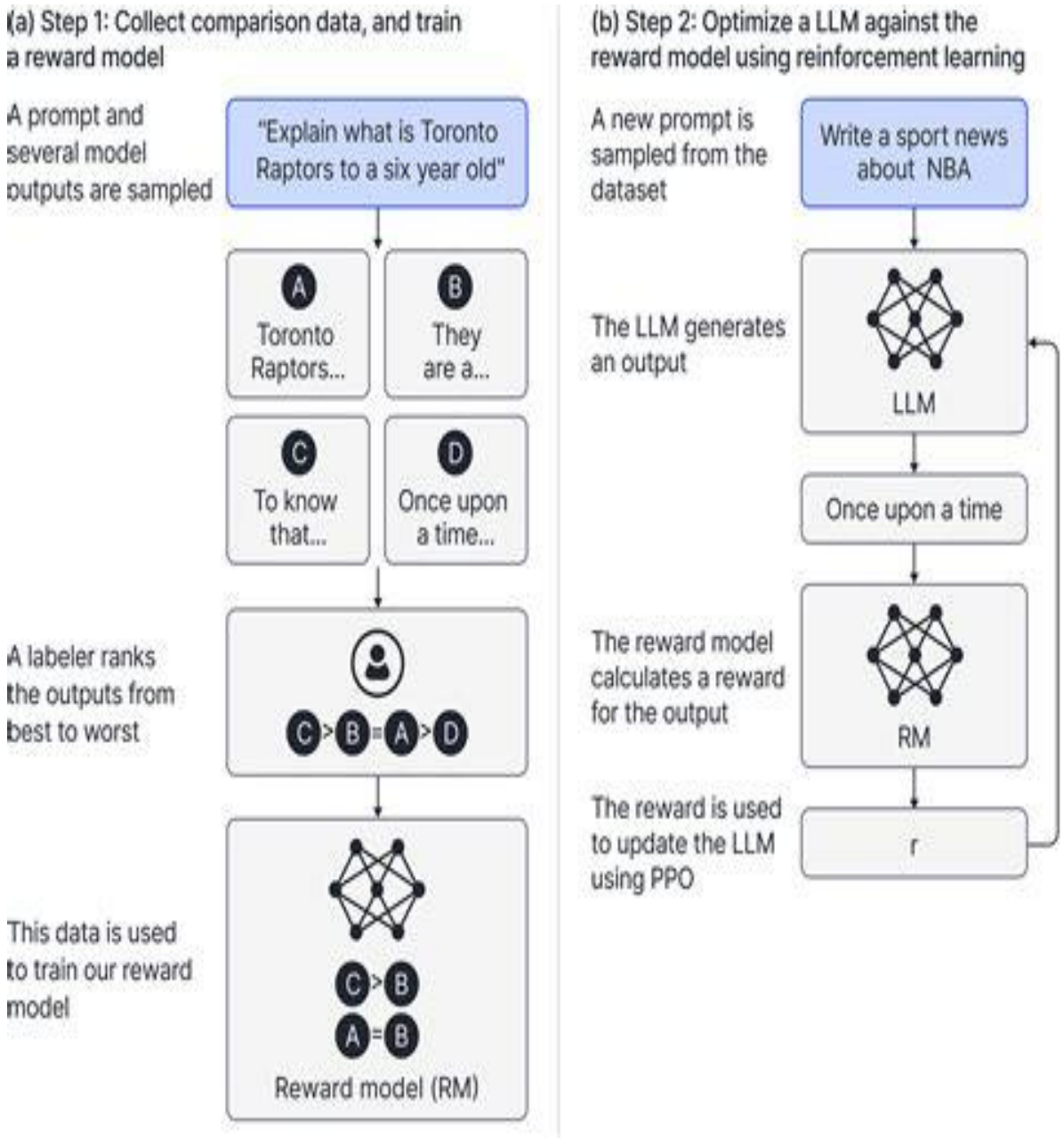


Dataset size	x00 billions to 1.x trillion tokens	~xk to x0k (prompt, response)	~x0k prompt
Example of models	GPT-3, LLaMA, Falcon, BLOOM	Dolly-v2, Falcon-instruct	Claude, GPT-4, ChatGPT
	(a)Pre-training	(b) Instruction fine-tuning	(c) Reinforcement learning from human feedback

LLM Reward Modelling

- Technique of *learning-to-rank*
  - To learn the relative rank order of responses

  - LLM generates N responses
  - A Human evaluator ranks them
  - LLM learns from the feedback



Direct Preference Optimization (DPO)

- It uses human annotator based prompt evaluation dataset
- Datasets contains 3 things
  - Prompt (user input)
  - Response (o/p of LLM)
  - Label : Like /dislike
- Based on like / dislike data by human annotators, LLM is trained to give responses which will be liked by humans
- DPO uses binary cross entropy loss just like classification task

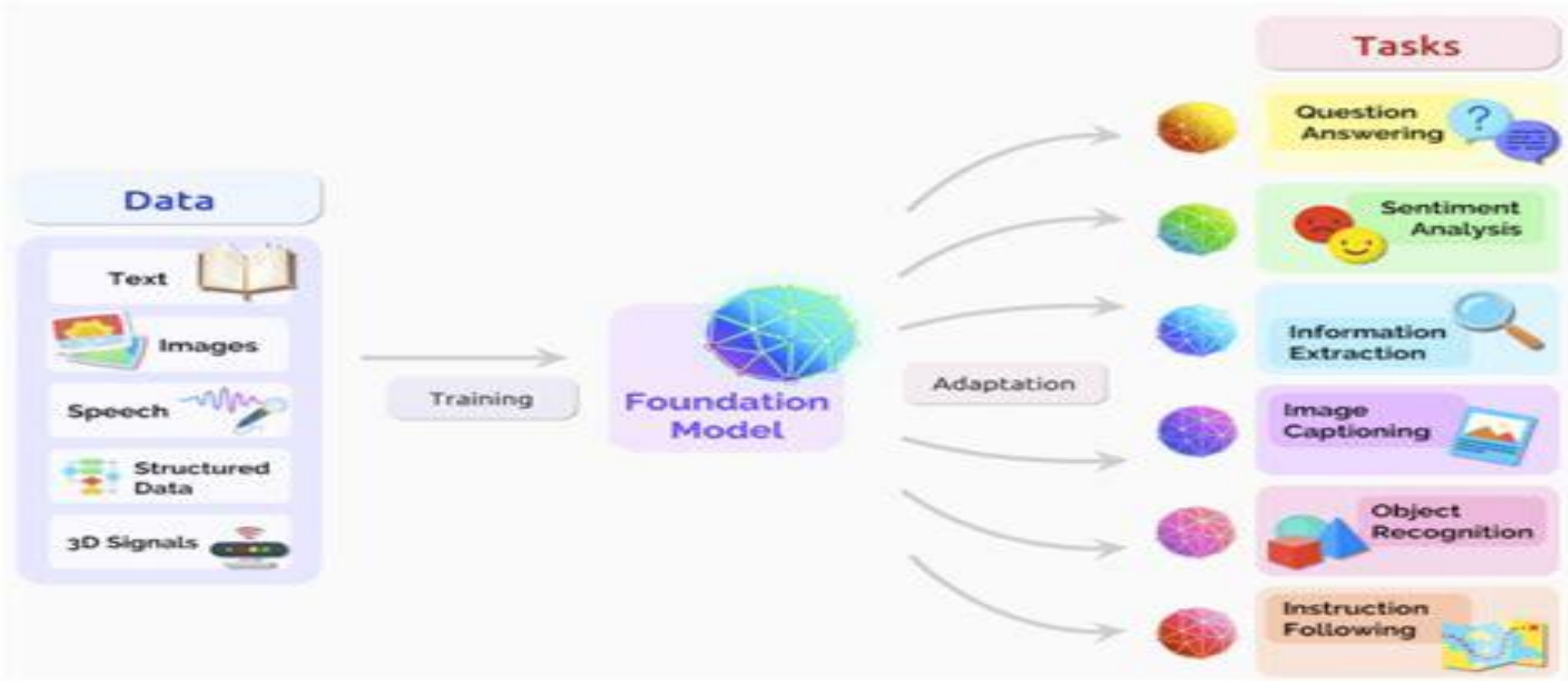
- Advantage :
- It doesn’t require extra LLM for reward modelling so training is cost effective

LLM Management Details

LLM Costs


appypie
Time and Iteration
Scalability and Infrastructure
Ethical and Environmental Considerations
Return on Investment (ROI)
The Computational Cost
Data Collection and Preprocessing
Human Expertise
The Cost Implications of Large Language Model (LLM) Training

LLM Applications



LLM Comparison

Comparing different LLMS

Model	Size	Use	Training code available	Inference code available	Finetuning code available	Code license	Weights license	Instruction-tuned / foundation model	Backbone
Bloomz	176B	Restricted applications	❌	✅	❌	Responsible AI (OpenRail)	Responsible AI (OpenRail)	Instruction-tuned	Bloom
Chat GPT (gpt-3.5-turbo)	175B	Paid API	❌	❌	❌	Public Web API	Public Web API	Instruction-tuned	GPT3
Dolly-V2	1B	Commercial	❌	❌	❌	Apache License 2.0	Apache License 2.0	Instruction-tuned	Pythia
Lit-LLaMA	7B	Non-commercial research	✅	✅	✅	Apache License 2.0	Non-commercial research	Foundation model	–
Lit-LLaMA + Alpaca	7B	Non-commercial research	✅	✅	✅	Apache License 2.0	Non-commercial research	Instruction-tuned	LLaMA

Evaluation metrics for ChaGPT / LLMs

Benchmarks	Description	Reference URL
GLUE Benchmark	GLUE (General Language Understanding Evaluation) benchmark provides a standardized set of diverse NLP tasks to evaluate the effectiveness of different language models	https://gluebenchmark.com/
SuperGLUE Benchmark	Compares more challenging and diverse tasks with GLUE, with comprehensive human baselines	https://super.gluebenchmark.com/
HellaSwag	Evaluates how well an LLM can complete a sentence	https://rowanzellers.com/hellaswag/
TruthfulQA	Measures truthfulness of model responses	https://github.com/sylinrl/TruthfulQA
MMLU	MMLU ((Massive Multitask Language Understanding) evaluates how well the LLM can multitask	https://github.com/hendrycks/test

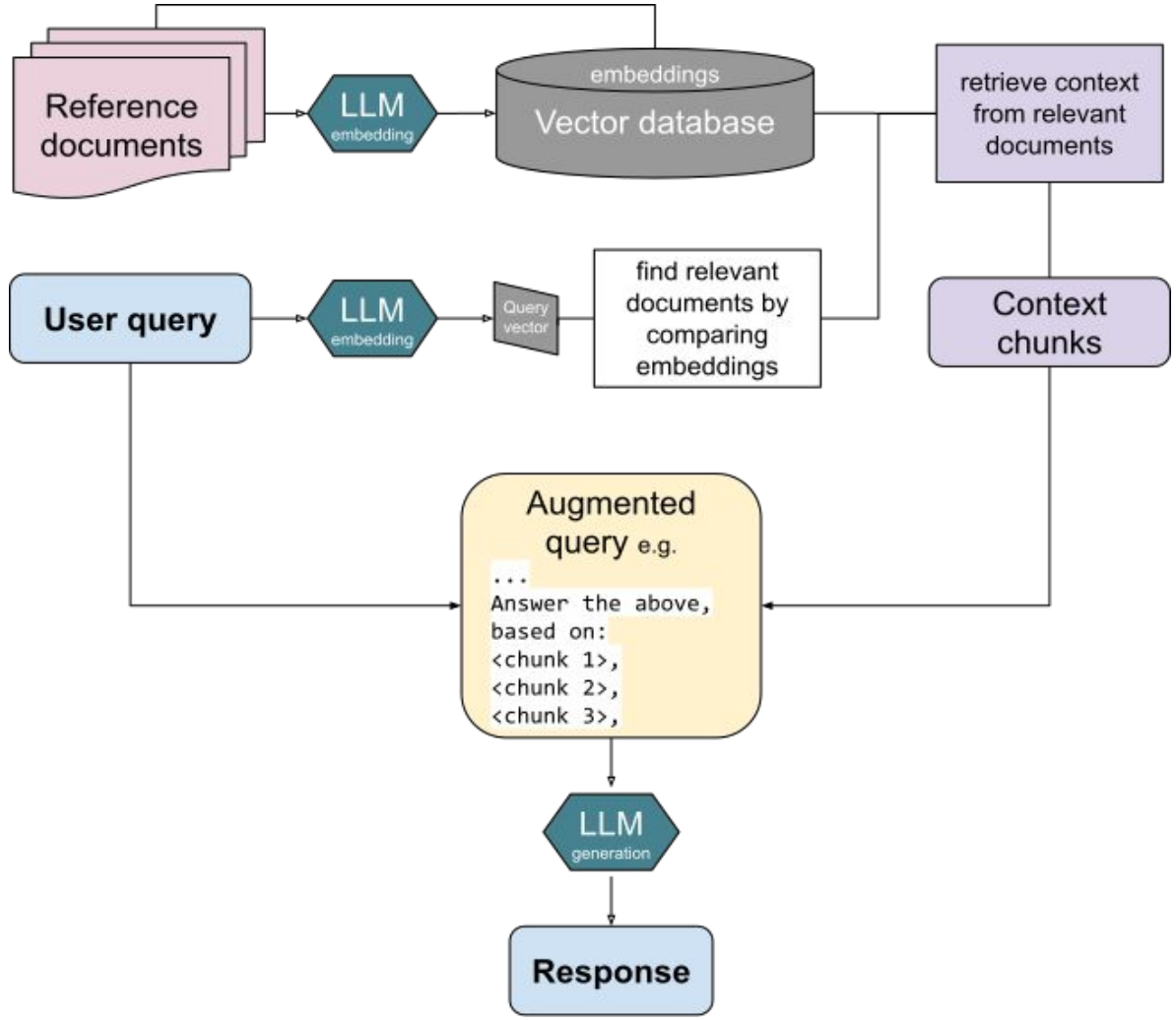
LLM Applications

RAG

Retrieval-augmented generation (RAG)

It helps generating responses based on ***data retrieved*** from reference documents stored in form of ***vector DB***

Future:
Multi-agent based systems



LLM Drawbacks

- Data bias
  - Data Privacy Issues
- Environmental impact
  - High carbon emissions
- Limited interpretability
  - Hallucination
  - Common Mistakes in generation
  - AI needs regulation
- Limited generalization
- Ethical concerns
  - Use for hacking / un-ethical purposes
- Attacks on LLMs

ChatGPT

AI-generated text in images can sometimes have spelling inconsistencies. To ensure perfect accuracy, you may want to create a word cloud manually using tools like **WordClouds.com**, **Canva**, or **WordArt.com**.

However, I can try generating a new version with clearer text. Would you like me to proceed?

yes

LLM Deployment Types

- Local Deployment
  - Great control over data
  - Control over responses
  - Low security
  - High efforts for compliance to standards
  - High investments
- Cloud Deployment
  - Advanced Security
  - Easy compliance to standards
  - Pay per use model
  - Less control on data
  - Less control on responses

References

- https://ar5iv.labs.arxiv.org/html/2304.08103
- https://rbcborealis.com/research-blogs/a-high-level-overview-of-large-language-models/
- https://medium.com/the-llmops-brief/introduction-to-large-language-models-9ac028d34732
- https://www.vellum.ai/llm-parameters/temperature
- https://muhtasham.github.io/blog/posts/llm-bootcamp/