Large Language Model

Can J.A.R.V.I.S becomes reality?

Author: occupymars

Date: June. 23, 2025

Version: 0.1

Contents

Chapter 1 Survey

1.1 Large Language Models: A Survey

Introduction

❑ *Early Pretrained Large Language Models*
❑ *Large LLMs*
❑ *LLM Techniques*
❑ *Datesets*
❑ *Benchmarks*
❑ *Challenges and Future*

More details please refre to Large Language Models: A Survey.

1.1.1 Introduction

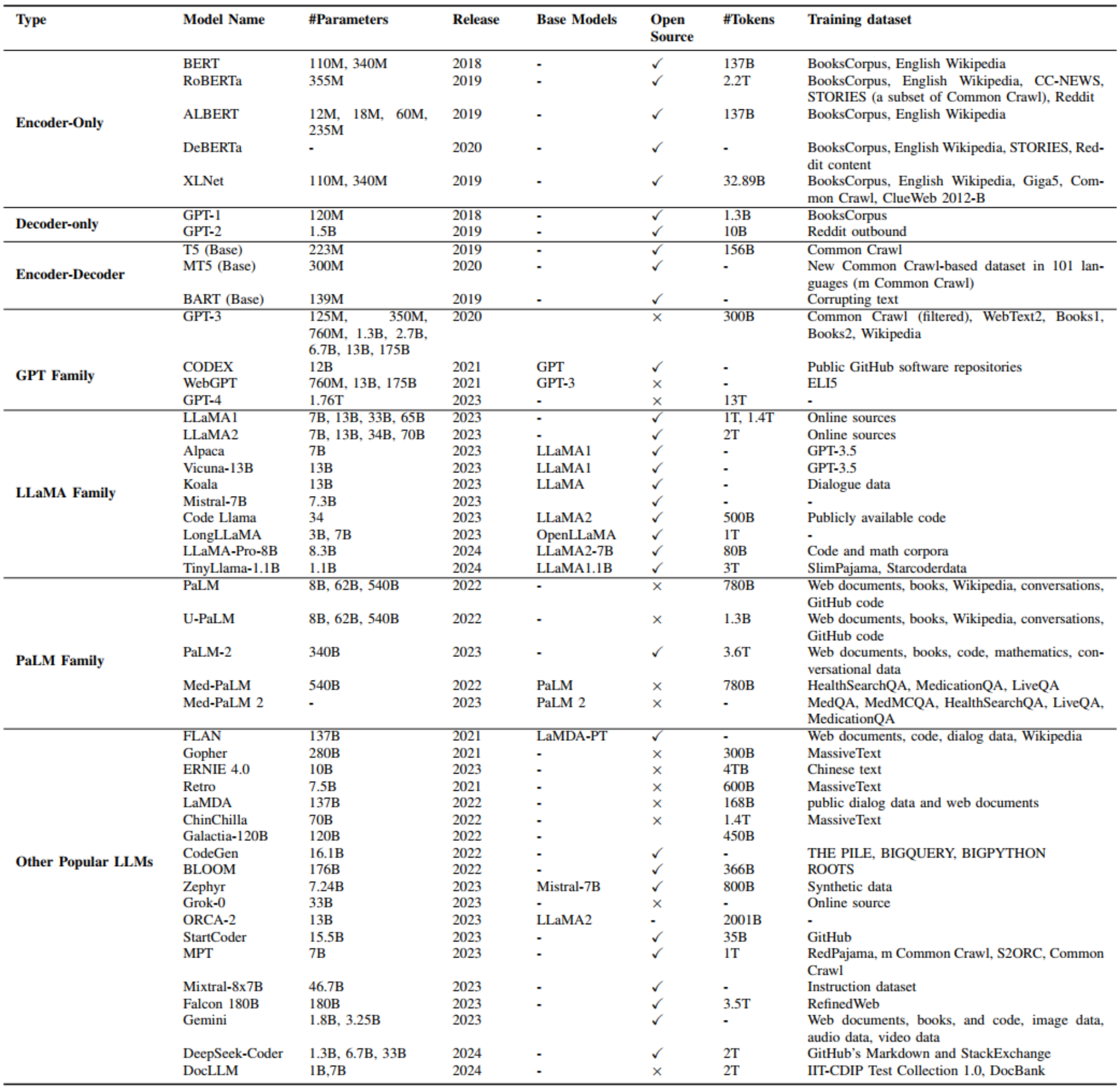
Statistical language model are not good because of the data sparsity, early neural language model are task specific.

Large language model have emergent abilities: ***in-context learning***, where LLMs learn a new task from a small set of examples presented in the prompt at inference time; ***instruction following***, where LLMs, after instruction tuning, can follow the instructions for new types of tasks without using explicit examples; ***multi-step reasoning***, where LLMs can solve a complex task by breaking down that task into intermediate reasoning steps.

1.1.2 Large Language Models

First let us walk through early pre-trained neural language models.

- The first category is encoder-only models, represented by BERT.
  1. BERT, pre-trained by two tasks: ***masked language modeling*** (MLM) and ***next sentence prediction*** (NSP)
  2. RoBERTa, remove NSP, train with much larger mini-batches and learning rates.
  3. ALBERT, from $V \times H$ to $V \times E \rightarrow E \times H$ by adding a linear layer, so the embedding layer will has a smaller param size. 24 layers are shared (but the computation speed is same). And new NSP, instead of randomly choose next sequence, inverse the true sequence pair order.
  4. DeBERTa (Decodingenhanced BERT with disentangled attention), use ***disentangled attention*** compute the content and position seperately; At fine-tune stage, add absolute position into decoding layer to predict masked tokens, enhance the MLM; Introduce a novel pre-training task, ELECTRA, known as ***replace token detection*** (RTD), instead of mask the word, we replace it with plausible alternatives sampled from a small generated model. And pre-training will try to distinguish whether tokens are replaced or not. RTD is more sample-efficient than MLM because the former is defined over all input tokens rather than just the small subset being masked out.
  5. XLMs, 3 pre-training method, casual language model (CLM) for warmup and basic ability; MLM, shared BPE between different language; Translation Language Modeling (TLM), [CLS] ENGLISH [SEP] FRANCH [SEP] with mask.
  6. XLNet, based on Transformer-XL, utilize auto-regressive (decoder), use PLM (permutation language model) to permutate the tokens of a sequence, use AR to predict it (only predict last K tokens) so it can get bidirection information, then use a two stream attention because PLM make the position a problem to handle.
  7. UNILM, shared Transformer but three pre-training method, AR, AE (encoder only), seq2seq (encoder-decoder), using different masks to control the context the prediction is conditioned on.
- Second category is decoder-only models, represented by GPT.
  1. GPT-1, next token prediction
  2. GPT-2, Layer normalization is moved to the input of each sub-block, additional layer normalization is added after the final self-attetion block.



Type	Model Name	#Parameters	Release	Base Models	Open Source	#Tokens	Training dataset
Encoder-Only	BERT	110M, 340M	2018	-	✓	137B	BooksCorpus, English Wikipedia
	RoBERTa	355M	2019	-	✓	2.2T	BooksCorpus, English Wikipedia, CC-NEWS, STORIES (a subset of Common Crawl), Reddit
	ALBERT	12M, 18M, 60M, 235M	2019	-	✓	137B	BooksCorpus, English Wikipedia
	DeBERTa	-	2020	-	✓	-	BooksCorpus, English Wikipedia, STORIES, Reddit content
	XLNet	110M, 340M	2019	-	✓	32.89B	BooksCorpus, English Wikipedia, Giga5, Common Crawl, ClueWeb 2012-B
Decoder-only	GPT-1	120M	2018	-	✓	1.3B	BooksCorpus
	GPT-2	1.5B	2019	-	✓	10B	Reddit outbound
Encoder-Decoder	T5 (Base)	223M	2019	-	✓	156B	Common Crawl
	MT5 (Base)	300M	2020	-	✓	-	New Common Crawl-based dataset in 101 languages (m Common Crawl)
	BART (Base)	139M	2019	-	✓	-	Corrupting text
GPT Family	GPT-3	125M, 350M, 760M, 1.3B, 2.7B, 6.7B, 13B, 175B	2020		×	300B	Common Crawl (filtered), WebText2, Books1, Books2, Wikipedia
	CODEX	12B	2021	GPT	✓	-	Public GitHub software repositories
	WebGPT	760M, 13B, 175B	2021	GPT-3	×	-	ELI5
	GPT-4	1.76T	2023	-	×	13T	-
LLaMA Family	LLaMA1	7B, 13B, 33B, 65B	2023	-	✓	1T, 1.4T	Online sources
	LLaMA2	7B, 13B, 34B, 70B	2023	-	✓	2T	Online sources
	Alpaca	7B	2023	LLaMA1	✓	-	GPT-3.5
	Vicuna-13B	13B	2023	LLaMA1	✓	-	GPT-3.5
	Koala	13B	2023	LLaMA	✓	-	Dialogue data
	Mistral-7B	7.3B	2023		✓	-	-
	Code Llama	34	2023	LLaMA2	✓	500B	Publicly available code
	LongLLaMA	3B, 7B	2023	OpenLLaMA	✓	1T	-
	LLaMA-Pro-8B	8.3B	2024	LLaMA2-7B	✓	80B	Code and math corpora
	TinyLlama-1.1B	1.1B	2024	LLaMA1.1B	✓	3T	SlimPajama, Starcoderdata
PaLM Family	PaLM	8B, 62B, 540B	2022	-	×	780B	Web documents, books, Wikipedia, conversations, GitHub code
	U-PaLM	8B, 62B, 540B	2022	-	×	1.3B	Web documents, books, Wikipedia, conversations, GitHub code
	PaLM-2	340B	2023	-	✓	3.6T	Web documents, books, code, mathematics, conversational data
	Med-PaLM	540B	2022	PaLM	×	780B	HealthSearchQA, MedicationQA, LiveQA
	Med-PaLM 2	-	2023	PaLM 2	×	-	MedQA, MedMCQA, HealthSearchQA, LiveQA, MedicationQA
Other Popular LLMs	FLAN	137B	2021	LaMDA-PT	✓	-	Web documents, code, dialog data, Wikipedia
	Gopher	280B	2021	-	×	300B	MassiveText
	ERNIE 4.0	10B	2023	-	×	4TB	Chinese text
	Retro	7.5B	2021	-	×	600B	MassiveText
	LaMDA	137B	2022	-	×	168B	public dialog data and web documents
	ChinChilla	70B	2022	-	×	1.4T	MassiveText
	Galactia-120B	120B	2022	-		450B	
	CodeGen	16.1B	2022	-	✓	-	THE PILE, BIGQUERY, BIGPYTHON
	BLOOM	176B	2022	-	✓	366B	ROOTS
	Zephyr	7.24B	2023	Mistral-7B	✓	800B	Synthetic data
	Grok-0	33B	2023	-	×	-	Online source
	ORCA-2	13B	2023	LLaMA2	-	2001B	-
	StartCoder	15.5B	2023	-	✓	35B	GitHub
	MPT	7B	2023	-	✓	1T	RedPajama, m Common Crawl, S2ORC, Common Crawl
	Mixtral-8x7B	46.7B	2023	-	✓	-	Instruction dataset
	Falcon 180B	180B	2023	-	✓	3.5T	RefinedWeb
	Gemini	1.8B, 3.25B	2023		✓	-	Web documents, books, and code, image data, audio data, video data
	DeepSeek-Coder	1.3B, 6.7B, 33B	2024	-	✓	2T	GitHub's Markdown and StackExchange
	DocLLM	1B,7B	2024	-	×	2T	IIT-CDIP Test Collection 1.0, DocBank

Figure 1.1: LLM Overview

- Last category is encoder-decoder PLMs, represented by T5.
  1. T5 and mT5 (multilingual variance of T5).
  2. MASS (MAsked Sequence to Sequence pre-training), mask several consecutive tokens as input, decoder predicts the masked fragment, so it jointly trains language embedding and generation.
  3. BART, it is pre-trained by corrupting text with an arbitrary noising function, and then learning to reconstruct the original text.

Then we comes to LLM family.

- GPT Family
  1. GPT-3, maybe the first LLM, with 175B parameters, the first to show in-context learning ability (one of the emergent ability).
  2. CODEX, based on GPT-3, fine-tuned on GitHub, powers Copilot.
  3. WebGPT, based on GPT-3, 3 steps fine-tuned to answer open-ended questions using a text-based web browser (mimic the human, reward, reinforce).
  4. InstructGPT, SFT on labeler-written prompt and demostrations, then further with human-ranked model ouputs build a reinforcement learning flow.
  5. ChatGPT, sibling (brother) model to InstructGPT, the Milestone.
  6. GPT-4, larger, multimodal input.
- LLaMA
  1. LLaMA-1, use the transformer architecture of GPT-3, with modifications: use SwiGLU instead of ReLU; use rotary positional embeddings instead of absolute one; use RMS layer normalization instead of standard one; 13B are better than GPT-3 175B.
  2. LLaMA-2, pre-train -> SFT -> RLFH, last step use accumulation of human feedback for revising the reward model (instead of a big change at short time) to save the ability of the model.
  3. Alpaca, first human create 175 instructions, then use pre-trained model to generate other instructions, then generate outputs, clean up, use as the SFT data, so this is efficient.
  4. Vicuna-13B, fine-tuned on user-shared conversations from ShareGPT.
  5. Guanaco, use ***QLoRA*** to fine-tune, back-propagates gradients through a frozen, 4-bit quantized pre-trained language model into Low Rank Adapters.
  6. Koala fine-tuned with data from ChatGPT.
  7. Mistral-7B leverages grouped-query attention for faster inference, sliding window attention to efficiently handle sequences of arbitrary length. Better than most 13B and 34B model at that time.
- PaLM (Pathways [a distrubuted system] Language Model)
  1. PaLM is a 540B model.
  2. U-PaLM trained with UL2R (Unified Language Learner with Residual Learning), first learn simple task (80% are simple tasks), then learn complicated one, use mixture-of-denoiser objective.
  3. Flan-PaLM is fine-tuned U-PaLM with instrunction. Large SFT datasets, 473 datasets with 146 categories.
  4. PaLM-2 is efficient and multilingual model.
  5. Med-PaLM, use ***instruction prompt tunning***, add a soft prompt at the embedding, and learn it (the model is frozen). So the method is more efficient than adapter method like LoRA, and it preformance is better than hard prompting.
- Other LLMs
  1. FLAN, shows that ***instruction fine-tune*** is good for zero-shot and multi-shot preformance, inspires ChatGPT instruction align tech and prompt engineer.
  2. Gopher, analysis of model scales on model preformance.
  3. T0, extend of FLAN, multi-prompt training.
  4. ERNIE 3.0, combine AR and AE. Injection of knowledge graph.
  5. RETRO, milestone of ***Retrieval-Augmented Generation***, with a similarity based chunk cross attention.
  6. GLaM, ***Mixture-of-experts*** (MOE), replace the FFN layer with Gated FFN, which will choose different sub-

layer using the gate, less training computation and inference computation.

7. LaMDA, trained on dialog data, focus on safety.
8. OPT
9. Chinchilla, limite the compute budget, train different size of model and dateset, find that these two size should be sacled equally.
10. Galactica, trained on scientific corpus.
11. CodeGen, conversational programming (multi-step programming).
12. AlexaTM, mixture of denoising and CLM tasks, more efficient few-shot learners thant decoder-only models on various tasks. Multilingual model.
13. Sparrow, use RLFH to create harmless, safety dialogue. Two modeling of outputs to help human labeler.
14. Minerva, math enhanced.
15. MoD, mixture-of-denoisers as the objectives.
16. BLOOM, which promotes transparent, collaborative AI research.
17. GLM, english-chinese model.
18. Pythia, trained on public data in a consistent sequence, with 154 checkpoints per model. It enables research on LLM training dynamics, memorization, term frequency effects, and gender bias reduction.
19. Orca, imitation learning, teacher assistance from ChatGPT.
20. StarCoder, 80+ programming languages.
21. KOSMOS, advancing research in cross-modal AI applications.

1.1.3 How LLMs are built

First we need to deal with data.

- Data Filtering: Removing the Noise, classifier-based, and heuristic-based frameworks. Handling Outliers. Addressing Imbalances. Text Preprocessing, removing stop words, punctuation or other elements that may not contribute significantly to the model's learning. Dealing with Ambiguities.
- Deduplication: remove duplicate instances, this is important becuase duplicated data will bring biases. The method for this can vary a lot, like fuzzy and exact deduplication.

Then we need to tokenize the text.

- BytePairEncoding
- WordPieceEncoding, start from characters (same as BPE), but when merging, maximize the likelihood of the language model rather than based only on frequncy.
- SentencePieceEncoding, above two assume blank space seperate words, this method start from unicode characters.

Let's we introduce positional encoding.

- ***Absolute Positional Embedding*** (APE), drawback is the restriction to a certain number of tokens, and does not count for relative distance.
- ***Relative Positional Embedding*** (RPE), add on keys and values.
- ***Rotary Position Embeddings***, used in LLaMA.
- ***Relative Positional Bias***, to facilitate extrapolation during inference.

Model Pre-training:

- Autoregressive Language Modeling, $\mathscr{L}_{ALM}(x) = \sum_{i=1}^{N} p(x_{i+n} \mid x_i, \dots, x_{i+n-1})$
- Masked Language Modeling, $\mathscr{L}_{MLM} = \sum_{i=1}^{N} p(\tilde{x} \mid x \setminus \tilde{x})$
- MoE

Fine-tuning and Instruction Tuning. Alignment.

- RLHF and RLAIF (reinforcement learning from AI feedback).

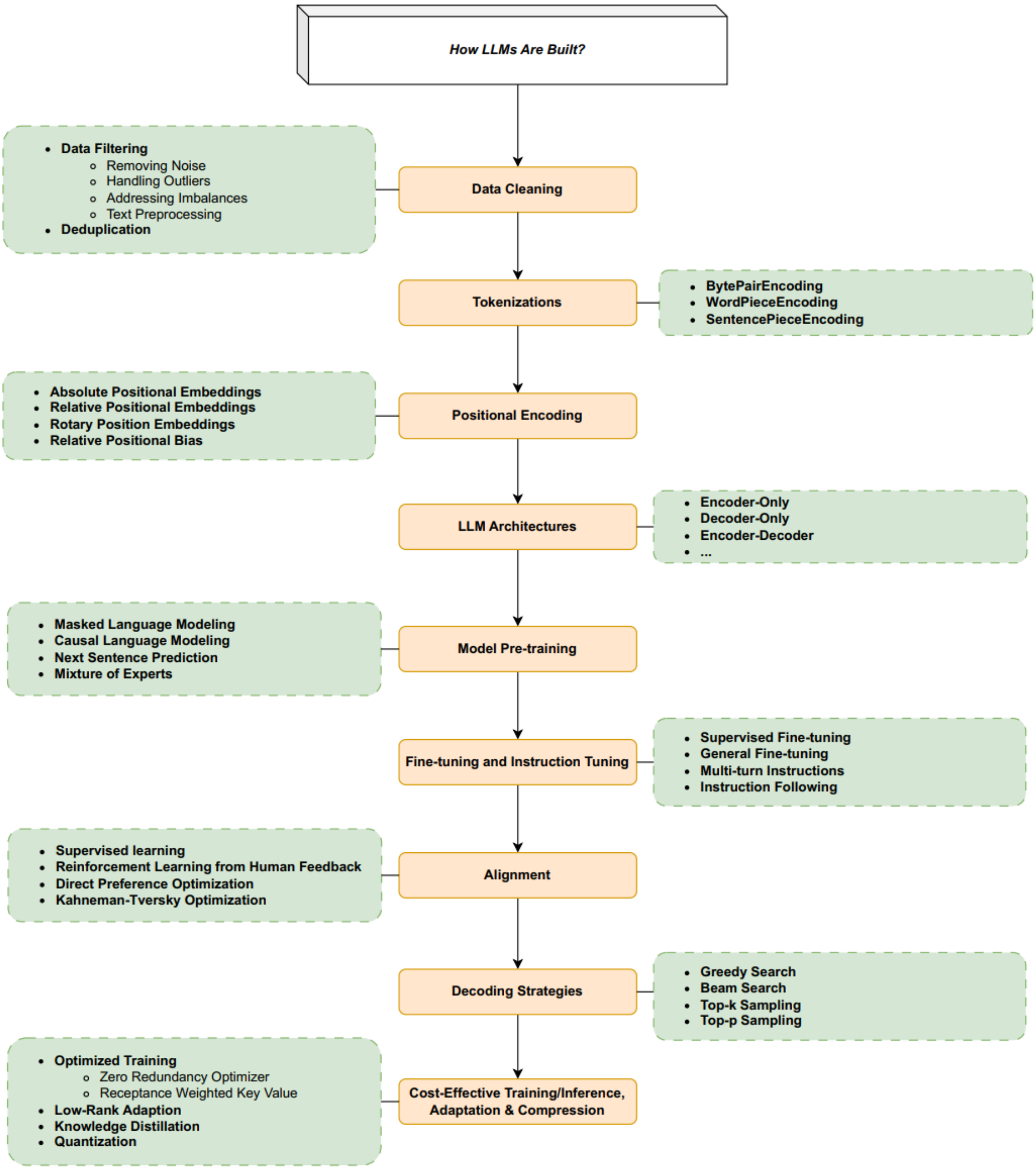


Figure 1.2: Components of LLMs

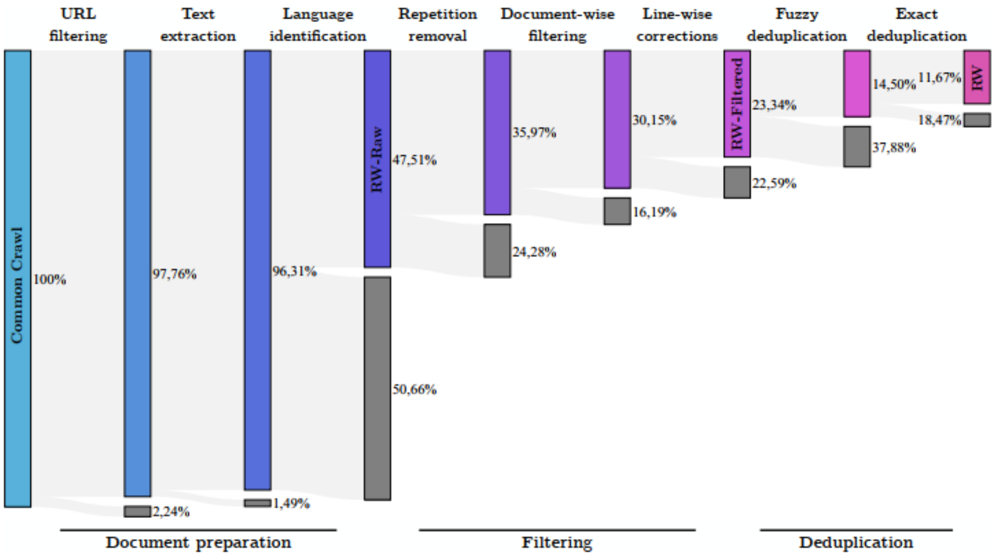


Figure 1.3: Data Preprocess

- DPO (Direct Preference Optimization), RLHF is complex and unstable. They leveraged a mapping between reward functions and optimal policies to show that this constrained reward maximization problem can be optimized exactly with a single stage of policy training, essentially solving a classification problem on the human preference data.
- Kahneman-Tversky Optimization (KTO), does not need paired preference date (x, y_w, y_l), only need (x, y) and knowledge of whether y is desirable or undersirable.

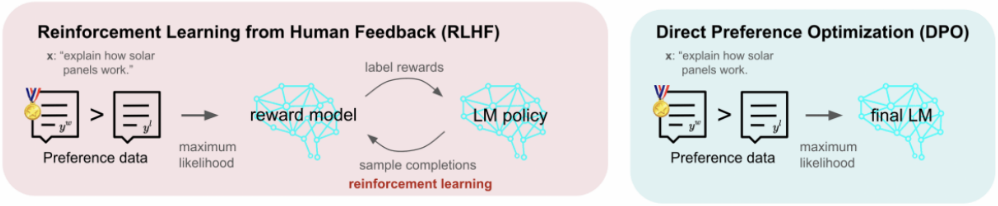

Reinforcement Learning from Human Feedback (RLHF)
x: "explain how solar panels work."
Preference data
maximum likelihood
reward model
label rewards
sample completions
LM policy
reinforcement learning
Direct Preference Optimization (DPO)
x: "explain how solar panels work.
Preference data
maximum likelihood
final LM

Figure 1.4: RLHF vs DPO

Decoding Strategies:

- Greedy Search, use the most probable token, does not consider the entire sequence.
- Beam Search, take N most likely tokens, repeate until max length, and take the maximum total score.
- top-k sampling, randomly choose one from k most likely options. $softmax(x_i) = e^{x_i/T} / \sum_j e^{x_j/T}$.
- top-p sampling, also known as Nucleus sampling, choose n tokens that $\sum_i^n p(t_i) > P$.

Then we come to cost-efficient training, inference, adaption, compression methods.

- Optimized training: Zero Redundancy Optimizer (ZeRO), data- and model-parallel. Receptance Weighted Key Value (RWKV) reformulates the attention mechanism to operate in a recurrent fashion, linear time complexity and constant space complexity with respect to sequence length T.
- Low-Rank Adaption (LoRA)
- Knowledge Distillation, response distillation (only mimic outputs), feature distillation (also intermediate layers), API distillation (similar to response distillation).
- Quantization, post training quantization and quantization-aware training.

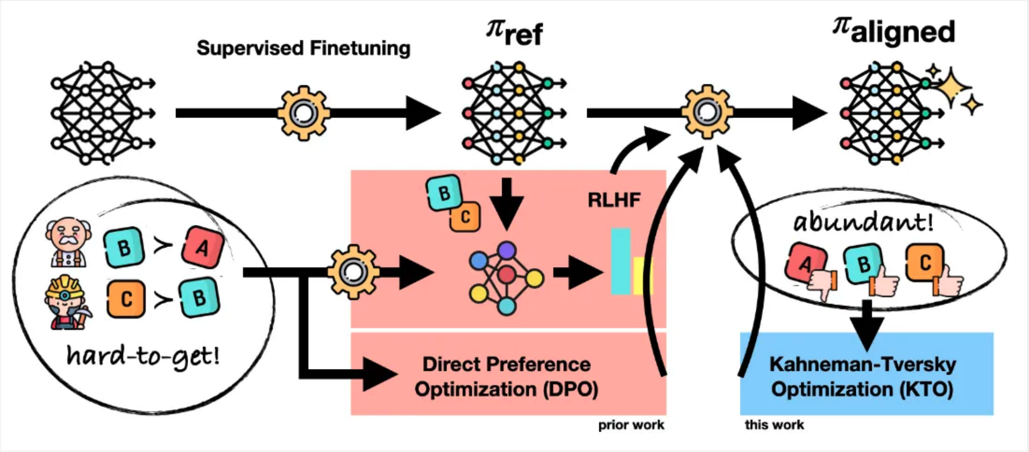


Figure 1.5: Alignment Methods

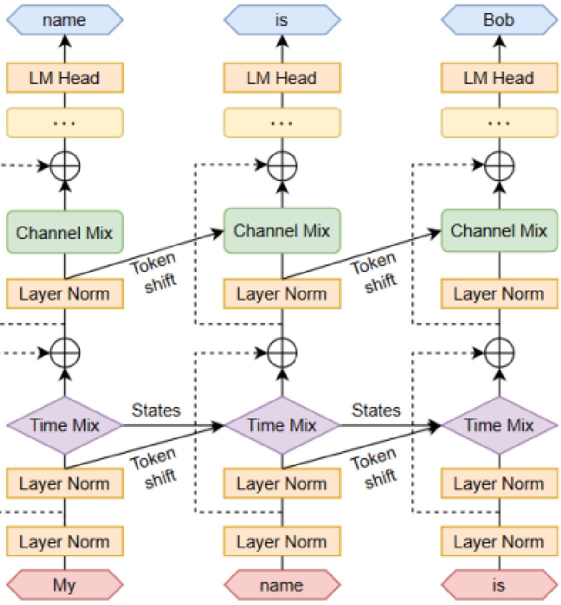


Figure 1.6: RWKV architecture

1.1.4 How LLMs are used and augmented

Hallucinations, categorized into instrinsic and extrinsic hallucination. We can use statistical and model-based matrices to measure it, for example, ROUGE, BLEU, PARENT, Knowledge F1 are statistical, IE-Based, QA-Based, NLI-Based and Faithfulness Classification matrics are model-based. And human scoring and comparative analysis is also vital. FactScore use both human and model-based, it breaks LLM generation into atomic facts. Ways to mitigate hallucination are Product Design and User Interaction Strategies, Data Management and Continuous Improvement, Prompt Engineering and Metaprompt Design, Model Selection and Configuration for Hallucination Mitigation.

Prompt Engineering approaches:

- ***Chain of Thought*** (CoT), described in *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*, there are Zero-Shot CoT (let LLM think step by step) and Manual CoT (give example that shows the reasoning steps, it is better than zero-shot).
- Tree of Thought (TOT), first branch out into multiple thought trees than converge, good for complex tasks.
- Self-Consistency, generate multiple output and check the consistency, if they are consistent, then these outputs are more likely accurate.
- Reflection, let LLM revise its answer.
- Expert Prompting, let LLM response as an expert or more experts (which is better).
- Chains, *PromptChainer: Chaining Large Language Model Prompts through Visual Programming*, second task depend on first one, create a chain.
- Rails, The concept of Rails involves setting up a framework or a set of guidelines that the LLM must follow while generating responses.
- Automatic Prompt Engineering (APE), let LLM to generate the prompts, evaluate them and refine them and iterates.

Retrieval Augmented Generation (RAG): Forward-looking Active Retrieval Augmented Generation (FLARE) is an advanced RAG that will check the confidence level of the LLM outputs, if it is low, the output will be used as a query to retrieve again, this is a iterative methods. More details of RAG please refer to *Retrieval-Augmented Generation for Large Language Models: A Survey*.

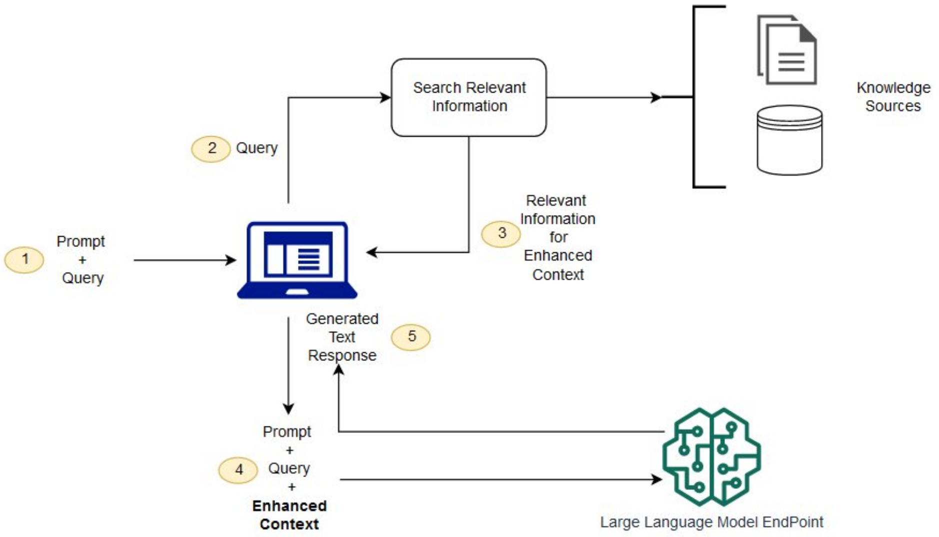

Search Relevant Information
Knowledge Sources
2 Query
1 Prompt + Query
3 Relevant Information for Enhanced Context
Generated Text Response 5
4 Prompt + Query + Enhanced Context
Large Language Model EndPoint

Figure 1.7: RAG

Then the using of external tools, *Toolformer: Language Models Can Teach Themselves to Use Tools* shows a demo. Automatic Multi-step Reasoning and Tool-use (ART) involves a systematic approache where, given a task and input, it will search from a task library for the similar task, use it as examples, we can also have tool library too.

LLM Agents: we introduce prompt engineering Techniques for agents, more information please refer to other surveys.

- Reasoning without Observation (ReWOO), the LLM initially develops a plan (a series of steps) that outlines how to approach and solve a given problem. The execution phase then involves integrating actual data or observations into the pre-specified plan, leading to coherent and contextually relevant responses. This method is good for costly, slow retrieval.
- Reason and Act (ReAct), the LLM is prompted to alternate between generating reasoning traces (explanations) and taking actions (steps or commands) in an interleaved manner.
- Dialog-Enabled Resolving Agents (DERA), is developed based on the idea of utilizing multiple agents within a dialog context, each with specific roles and functions.

1.1.5 Popular Datasets

These datasets are 1.8, and in the paper there is a table for the evaluation metric and links.

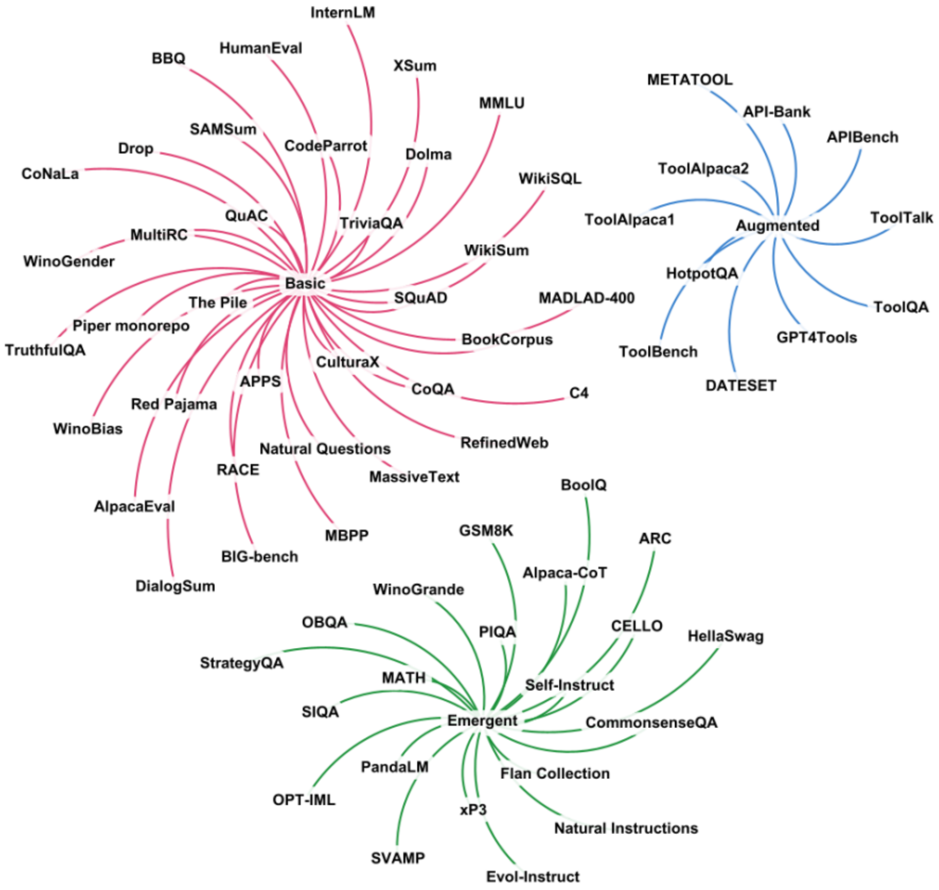


Figure 1.8: LLM Datesets

Evaluation, generate n solution, and if c pass:

$$\text{pass } @k = \mathbb{E}_{\text{problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]$$

Exact Match, must strictly match the true tokens than it returns 1 else returns 0. ***Human equivalence score*** (HEQ) compare LLM with average human accuracy.

1.1.6 Challenges and Future

1. Smaller model like Phi from Microsoft, more efficient training like parameter-efficient fine-tuning (PEFT).
2. New post-attention paradigms, like ***State Space Models*** (SSMs)
3. Multi-modal LLM
4. Safe LLM
5. Agents

Chapter 2 History Language Models

2.1 Statistical Language Model

N-Gram model is the most dominating form of SLMs, which is a Markov chain model, using the statistic from previous $n-1$ words and current word.

$$P(w_n \mid w_{n-k}, \ldots, w_{n-1}) = \frac{count(w_{n-k}, \ldots, w_n)}{count(w_{n-k}, \ldots, w_{n-1})}$$

in n-gram, k is always $n-1$.

In order to deal with unseen words and small probability of larger n-pairs, smooth method are often used, like ***Laplace-smooth***.

Chapter 3 LLM Techniques

3.1 Model Architecture

3.2 Dataset

3.3 Tokenize

3.4 Positional Encoding

Relative Positional Embedding, the attention score is:

$$e_{ij} = \frac{(q_i + r_{i-j})k_j}{\sqrt{d_k}}$$

where r_{i-j} is an embedding output. Another variants add this to the key or the QK matrix or even to the value matrix.

Rotary Position Embeddings, let use R for this operation, q_p, k_p are query vector and key vector at position p, then attention score is:

$$\text{score}(q_p, k_p) = \langle R(q_p, p), R(k_p, p) \rangle$$

And R has property: $\langle R(q, p), R(k, p^{'}) \rangle = \langle R(q, 0), R(k, p^{'} - p) \rangle$.

The math: given $x = [(x_1, x_2), \ldots, (x_{d-1}, x_d)]$, each pair is rotated by an angle $\theta_i \cdot p$, p is the position. Let $z_i = x_{2i} + ix_{2i+1}$, $R(z_i, p) = z_i \cdot e^{i\theta_i p}$. The matrix form is:

$$\begin{bmatrix} x^{'}_{2i} \\ x^{'}_{2i+1} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i p) & -\sin(\theta_i p) \\ \sin(\theta_i p) & -\cos(\theta_i p) \end{bmatrix} \begin{bmatrix} x_{2i} \\ x_{2i+1} \end{bmatrix}$$

RoPE can extrapolate to unseen longer sequences.

Relative Positional Bias, the math is:

$$e_{ij} = \frac{q_i k_j}{\sqrt{d_k}} + b_{i-j}$$

for the extrapolation during inference.

3.5 Pre-training

Low-Rank Adaption (LoRA), $W_0 + \Delta W = W_0 + BA$, where initializations are $A \sim \mathcal{N}(0, \sigma^2), B = 0$. Normally only attention weight will be adapted, the original weights are frozen.