Phase 1

SYSTEMS FOR MACHINE LEARNING

EE 599

Authors:

Hengyi Tang (8016801929)
Caoyi Zou (2234578518)

April 26, 2024

Contents

1 Background

Comprehensive questions from LLM survey paper and LLaMA papers. [**?**].

1. What are the four major aspects of LLMs covered in the LLM survey paper?

A: i) pre-training (how to pre-train a capable LLM)

ii) adaptation (how to effectively adapt pre-trained LLMs for better use)

iii) utilization (how to use LLMs for solving various downstream tasks)

iv) capability evaluation (how to evaluate the abilities of LLMs and existing empirical findings)

2. What are the three major differences between LLMs and PLMs? What are the three typical emergent abilities for LLMs?

A: 1) The three major differences between LLMs and PLMs:

First, LLMs display some surprising emergent abilities that may not be observed in previous smaller PLMs. These abilities are key to the performance of language models on complex tasks, making AI algorithms unprecedently powerful and effective.

Second, LLMs would revolutionize the way that humans develop and use AI algorithms. Unlike small PLMs, the major approach to accessing LLMs is through the prompting interface (e.g., GPT-4 API). Humans have to understand how LLMs work and format their tasks in a way that LLMs can follow.

Third, the development of LLMs no longer draws a clear distinction between research and engineering. The training of LLMs requires extensive practical experiences in large-scale data processing and distributed parallel training. To develop capable LLMs, researchers have to solve complicated engineering issues, working with engineers or being engineers.

2) The three typical emergent abilities for LLMs:

In-context learning. Assuming that the language model has been provided with a natural language instruction and/or several task demonstrations, it can generate the expected output for the test instances by completing the word sequence of input text, without requiring additional training or gradient update.

Instruction following. By fine-tuning with a mixture of multi-task datasets formatted via natural language descriptions (called instruction tuning), LLMs are shown to perform well on unseen tasks that are also described in the form of instructions. With instruction tuning, LLMs are enabled to follow the task instructions for new tasks without using explicit examples, thus having an improved generalization ability.

Step-by-step reasoning. For small language models, it is usually difficult to solve complex tasks that involve multiple reasoning steps, e.g., mathematical word problems. In contrast, with the chain-of-thought (CoT) prompting strategy, LLMs can solve such tasks by utilizing the prompting mechanism that involves intermediate reasoning steps for deriving the final answer.

3. Where are pre-training data from?

A: Existing LLMs mainly leverage a mixture of diverse public textual datasets as the pre-training corpus. The source of pre-training corpus can be broadly categorized into two types: general data and specialized data.

General data, such as webpages, books, and conversational text, is utilized by most LLMs due to its large, diverse, and accessible nature, which can enhance the language modeling and generalization abilities of LLMs.

In light of the impressive generalization capabilities exhibited by LLMs, there are also studies that extend their pre-training corpus to more specialized datasets, such as multilingual data, scientific data, and code, endowing LLMs with specific task-solving capabilities.

4. What are the three main types of instruction tuning datasets? What is the Alpaca dataset [**?**]? Pick a training sample and describe it.

A: The three main types of instruction tuning datasets:

NLP Task Datasets. This kind of datasets are formatted based on collected NLP task datasets (e.g., text classification and summarization) with corresponding natural language task descriptions.

Daily Chat Datasets. This kind of datasets are constructed based on real user conversations where queries are posed by humans and responses are mainly generated by human labelers or LLMs (e.g., ChatGPT, GPT-4).

Synthetic Datasets. This kind of datasets are typically constructed by instructing LLMs, based on pre-defined guidance rules or methods.

Alpaca is a synthetic dataset based on the selfinstruct method. It utilizes the text-davinci-003 model on the 175 seed datasets from Self-Instruct-52K to obtain 52,000 new instructions and corresponding inputs and outputs. Moreover, 60% of the examples are pure instructions without the input part in the final dataset.

5. What are the pertaining data used by LLaMA? How many tokens are in the entire training set for training LLaMA?

A: The pre-training dataset used by LLaMA is a mixture of several sources, list as follows in descending order: CommonCrawl [67%], C4[15%], GitHub[4.5%], Wikipedia[4.5%], Gutenberg and Books3[4.5%], ArXiv[2.5%], Stack Exchange[2%]. The training data size for LLaMa(6B) and LLaMa(13B) is 1.0 Tokens; while for LLaMa(32B) and LLaMa(65B) is 1.4 Tokens.

6. Before pre-training, what is the procedure for data preprocessing?

A: (i) Quality Filtering (ii) De-duplication (iii) Privacy Reduction (iv) Tokenization

7. What is tokenization? Name a few tokenization algorithms. What is the tokenization method used by LLaMA?

A: Tokenization aims to segment raw text into sequences of individual tokens, which are subsequently used as the inputs of LLMs.

Recently, subword tokenizers have been widely used in Transformer based language models, typically including Byte-Pair Encoding tokenization, WordPiece tokenization and Unigram tokenization.

LLaMa tokenize the data with the byte-pair encoding (BPE) algorithm, using the implementation from Sentence-Piece.

8. What are the main three types of transformer architecture? Name a few models for each type.

A: (i) Encoder-decoder Architecture: T5, BART, Flan-T5

(ii) Causal Decoder Architecture: GPT-series, OPT, PaLM, LLaMA, BLOOM, Gopher

(iii) Prefix Decoder Architecture: GLM130B and U-PaLM

9. Why are LLMs mainly decoder-only architecture?

A: The essence of decoder-only architecture is to accurately predict the next word for reconstructing the pre-training data. Since most language tasks can be cast as the prediction problem based on the input, these decoder-only LLMs might be potentially advantageous to implicitly learn how to accomplish these tasks in a unified LM way. Some studies have also revealed that decoder-only LLMs can be naturally transferred to certain tasks by autoregressively predicting the next tokens, without fine-tuning.

10. What is position embedding? Name a few position embedding algorithms.

A: Since the self-attention modules in Transformer are permutation equivariant, position embeddings (PE) are employed to inject absolute or relative position information for modeling sequences.

Position embedding algorithms: Absolute position embedding, Relative position embedding, Rotary Position Embedding, ALiBi

11. What is language modeling? What is the difference between causal language modeling and masked language modeling?

A: The language modeling task (LM) is the most commonly used objective to pre-train decoder-only LLMs, e.g., GPT3 and PaLM. Given a sequence of tokens x = x1, . . . , xn, the LM task aims to autoregressively predict the target tokens xi based on the preceding tokens x<i in a sequence.

12. How to generate text from LLMs [**?**]? Explain different decoding strategies.

A:

13. What is instruction tuning, and why is it important?

A: Instruction Tuning is the approach to fine-tuning pre-trained LLMs on a collection of formatted instances in the form of natural language, which is highly related to supervised fine-tuning and multi-task prompted training.

After instruction tuning, LLMs can demonstrate superior abilities to generalize to unseen tasks, even in multilingual setting.

14. What is alignment tuning, and why is it important?

A: Alignment Tuning is proposed to make LLMs act in line with human expectations. Such an alignment requires considering very different criteria, like helpfulness, honesty and harmlessness. It has been shown that alignment might harm the general abilities of LLMs to some extent, which is also called alignment tax in related literature.

However, such alignment can avert unexpected or unintended behaviors, e.g., fabricating false information, pursuing inaccurate objectives, and producing harmful, misleading and biased expressions.

15. What is parameter-efficient fine-tuning, and why is it important? What is the difference between prefix tuning and prompt tuning?

A: Since LLMs consist of a huge amount of model parameters, it would be costly to perform full-parameter tuning, thus parameter-efficient fine-tuning aims to reducing the number of trainable parameters while retaining good performance as possible.

Prefix tuning prepends a sequence of prefixes, which are a set of trainable continuous vectors, to each Transformer layer in language model; while prompt tuning mainly focus on incorporating trainable prompt vectors at the input layer.

16. What is prompt engineering, and why is it important? What is zero-shot and few-shot demonstrations?

A: The process of manually creating a suitable prompt is called prompt engineering, which is a major approach to utilizing LLMs for solving various tasks.

A well-designed prompt is helpful to elicit the ability of LLMs for accomplishing certain tasks.

Zero-shot: Provide a textual description of the task and a test example. The model either provides an answer using open-ended generation, or ranks the proposed answers.

Few-shot: Provide a few examples of the task (1-64) and a test example. The model takes this text as input and generates the answer or ranks different options.

17. What is the difference between in-context learning and chain-of-thought prompting?

A: ICL uses a formatted natural language prompt, consisting of the task description and/or a few task examples as demonstrations. First, starting with a task description, a few examples are selected from the task dataset as demonstrations. Them, they are combined in a specific order to form natural language prompts with specially designed templates. Finally, the test instance is appended to the demonstration as the input for LLMs to generate the output.

Instead of simply constructing the prompts with input-output pairs like ICT, CoT prompting further incorporates intermediate reasoning steps, which serve as the bridge between inputs and outputs.

18. What are the three basic types of ability evaluation for LLMs? What is perplexity [**?**]? What is hallucination?

A: 3 basic types of ability evaluation: Language Generation, Knowledge Utilization, Complex Reasoning.

Perplexity (PPL) is one of the most common metrics for evaluating language models. Perplexity is defined as the exponentiated average negative log-likelihood of a sequence. If we have a tokenized sequence $X = (x_0, x_1, ...x_t)$, then the perplexity of X is,

$$PPL(X) = exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\} \tag{1}$$

Hallucination: In generating factual texts, a challenging issue is hallucination generations, where the generated information is either in conflict with the existing source (intrinsic hallucination) or cannot be verified by the available source (extrinsic hallucination),

19. What is human alignment, and why is it important?

A: Human alignment is one of superior abilities that require special considerations for evaluation. It shows the ability that LLMs could well confrom to human values and needs, which represents a key ability for the broad use of LLMs in real world application.

20. Name a few comprehensive evaluation benchmarks for the evaluation of LLMs.

A: MMLU, BIG-bench, HELM, and a series of Human-level test benchmarks.

2 Optimization Methods

2.1 Gradient Accumulation

1. Consider a linear model with MSE loss, and mathematically explain why this division step can yield identical results.

A: Each loss value is the average of the 8 images small batch. When we accumulate the loss value of all 4 batches, then divide by 4 and get the MSE loss, which is equivalent to accumulating the loss value of all the images and devide by 32.

2. Mathematically explain the batch normalization layer behavior and why gradient accumulation yields non-identical results in this case.

A: Batch Normalization calculates the mean and variance of the samples and then use these statistics to normalize the data, which is done to reduce the internal covariate shift.

However, when gradient accumulation is applied, these statistics are calculated independently for the single batch, instead of the whole dataset. As a result, the mean and variance could be different, especially when the data distributions are significantly different from one small batch to another.

2.2 Gradient Checkpointing

1. Explain why model inference does not have such "memory blow-up" problem.

A: When executing inference, the trained model will compute predicted results directly, so it does not need to store all activations and intermediate results from the forward pass for performing backward propagation. This process does not have too many memory requirements.

Gradient checkpointing mitigates this problem by selectively storing certain activation values, called checkpoints. When activation values are needed later during backpropagation, the algorithm can recalculate them by performing a forward pass from the last checkpoint, instead of storing all activation values in memory throughout the training process. This tradeoff allows for a reduction in memory usage, but at the cost of adding additional computation because some activation values are recomputed rather than retrieved directly from memory.

2. What are the memory requirement and forward computation steps for the two strategies in big O notation?

A: daodoshisheizaihui

Poor strategy: memory: $O(n)$, computation: $O(n^2)$

General strategy: memory: $O(n^{3/2})$, computation: $O(n)$

2.3 Low-Rank Adaptation (LoRA)

1. What is matrix rank?

A: The dimension of the vector space generated (or spanned) by its columns (rows). This corresponds to the maximal number of linearity independent columns (rows) of A.

2. What are three decomposed matrices by SVD?

A: The SVD is a unique mateix decomposition that exists for every complex valued matrix $X \in \mathbb{C}^{m \times n}$:

$$X = U \Sigma V^* \tag{2}$$

where $U \in \mathbb{C}^{n \times n}$ and $V \in \mathbb{C}^{m \times m}$ are complex unitary matrices, and $\sum \in \mathbb{C}^{n \times m}$ is a rectangular diagonal matrix with non-negative entries on the diagonal and zeros off the diagonal. Here * denotes the complex conjugate transpose.

3. U and V are orthogonal matrices. Why does it imply $UU^T = I, VV^T = I$?

A: See transcript below.

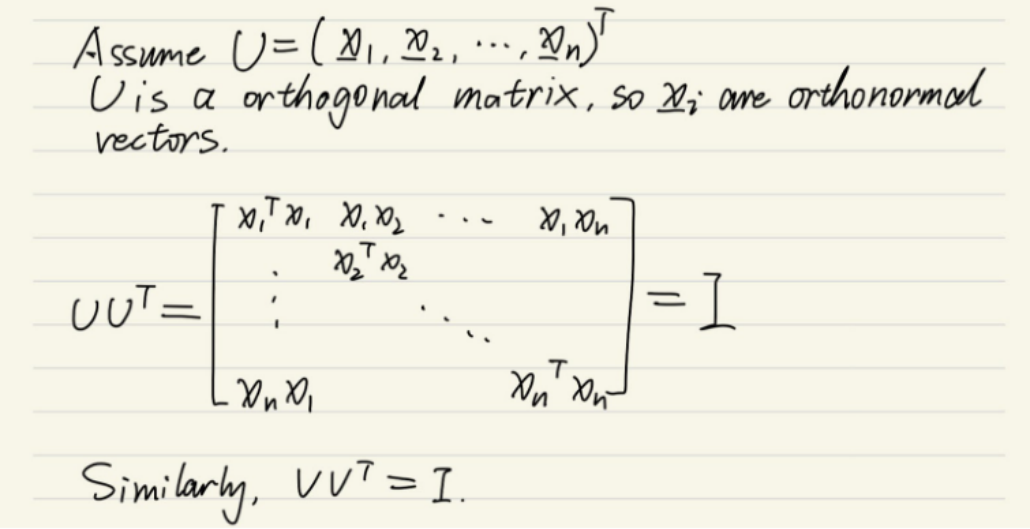

Figure 1: orthogonal matrix prove

4. If a matrix $W \in R^{n \times n}$ is full rank, what is its rank?

A: The diagonal will be filled with non-negative entries and no zeros off the diagonal. Thus the rank would be exactly n.

5. Suppose a full rank matrix $W \in R^{n \times n}$ represents an image. After we apply SVD to this matrix, we modify the singular matrix by only keeping its top-k singular values and discarding the rest (i.e., set the rest of the singular values to zero). Then, we reconstruct the image by multiplying U, modified S, and V. What would the reconstructed image look like? What if you increase the values of k (i.e., keep more singular values)?

A: When k is small: The reconstructed image $X(A)$ would only capture the most significant 'features' of the original image. These features correspond to the directions of greatest variance in the image data. The image might be blurry or missing details.

When increasing k : The approximation image $X(A)$ becomes more accurate, sharper and closer to the original as more of the higher ranked information (which corresponds to details and edges) is retained.

6. If a matrix $W \in R^{n \times n}$ is low rank, what does its singular matrix look like?

A: That means there would be r non-negative entries on the diagonal and $n - r$ zeros off the diagonal. Thus the rank would be r. Shown in graph Figure 2, source [SVD].

Figure 2: Singular Value Decomposition (SVD)

7. If the top-k singular values of a matrix $W \in R^{n \times n}$ are large, and the rest are near zero, this matrix W exhibits low-rank or near-low-rank behavior. Can you represent W by two low-rank matrices, A and B? If

so, what are those two matrices' expressions in terms of U, S, and V? Do you think those two matrices are a good approximation of W (i.e., $W \approx AB$)?

A: Let $\sum_k$ be the diagonal matrix of size $k \times k$ that contains only top-k singular values of matrix $W \in \mathbb{R}^{n \times n}$, Define U_k and V_k as the first k columns of U and V, respectively.

Then we may Set

$$A = U_k \sum_k^{1/2} \tag{3}$$

$$B = \sum_k^{1/2} V_k \tag{4}$$

where $\sum_k^{1/2}$ is the square rot of the diagonal matrix $\sum_k$, so each entry on the diagonal is the square root of the corresponding singular value.

It turns out that

$$AB = (U_k \sum_k^{1/2})(\sum_k^{1/2} V_k) = U_k \sum_k V_k^* \tag{5}$$

Using two low-rank matrices A and B for representation is efficient, particularly when the spectral energy of original image is concentrated in the first k singular values.

8. The above operation is called truncated SVD. Under what situation do you think truncated SVD fails to make a good approximation? Think about the singular matrix.

A: If singular values do not decay quickly, i.e., beyond the top-k values, the remaining are still significant, then truncating might lead to a substantial loss of information.

9. It seems like applying LoRA would add more parameters to the model, thus increasing forward pass costs and inference latency. How does LoRA paper overcome this drawback?

A: When deployed in production, we can explicitly compute and store $W = W_0 + BA$ and perform inference as usual. Note that both W_0 and BA are in $Rd \times k$. When we need to switch to another downstream task, we can recover W_0 by subtracting BA and then adding a different $B'A'$, a quick operation with very little memory overhead. Critically, this guarantees that we do not introduce any additional latency during inference compared to a fine-tuned model by construction.

2.4 Mixed Precision Training

1. Read the paper and answer why we need an FP32 master copy of weights and why we need to scale up the loss.

A: In mixed precision training, while using FP16 may speed up computation and reduces memory consumption, FP16 has limited range and accuracy for representation, which could lead to problems like information loss during weight updates. As a result, Maintaining a copy of the weights in FP32 format can ensure the accuracy throughout the training process, as FP32 provides a larger range of vaules and more accurate representation capabilities.

2. Do you think this statement still holds for LLM fine-tuning?

A: In the context of fine-tuning large language models (LLMs), this statement should largely hold true. Fine-tuning often involves adjusting a pre-trained model to a specific task or dataset, which can benefit from the speed and efficiency gains of mixed precision training. Since LLMs can be quite large, and the cost of storing activations is substantial, using mixed precision to store activations in FP16 can indeed halve the overall memory consumption during training, as the activations take up the majority of memory. Moreover, as the fine-tuning process does not usually require the full retraining of the model, the reduced precision of FP16 is often sufficient for the incremental updates made during fine-tuning.

References

[SVD] Image compression with singular value decomposition.