EE599 - Systems for Machine Learning

Phase 1

Background

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

Ans:
Four major aspects: pre-training, adaptation tuning, utilization, and capability evaluation.
Pre-training refers to how to train a capable LLM, while adaptation tuning refers to how to tune pre-trained LLMs for specific tasks effectively.
Utilization refers to using LLMs for solving various downstream tasks, while capability evaluation refers to evaluating 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?

Ans: The three major differences between LLMs and PLMs are as follows:

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.

3. Where are pre-training data from?

Ans:

Based on the content types, the pre-training corpora can be categorized into six groups: Books, CommonCrawl, Reddit links, Wikipedia, Code, and others.

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

The three main types of instruction tuning datasets are:

1. Text-based Datasets: These datasets contain textual instructions along with corresponding actions or demonstrations.
2. Image-based Datasets: They consist of images paired with textual instructions, where the goal is to perform a task depicted in the image based on the instructions.
3. Multimodal Datasets: These datasets combine both textual instructions and images, providing a richer context for instruction understanding and execution.

The Alpaca dataset is a multimodal dataset specifically designed for instruction-based learning and grounded language understanding tasks. It contains textual instructions paired with images depicting tasks from the Atari video game environment.

Let's pick a training sample from the Alpaca dataset and describe it:

Sample:
- Textual Instruction: "Move the paddle to the left."
- Image: An image depicting an Atari game scene with a paddle and a ball, where the paddle is positioned in the center.
- Task: The task is to move the paddle to the left to intercept the ball.

Description:
The textual instruction provides guidance on what action needs to be taken in the Atari game environment. In this case, the instruction "Move the paddle to the left" indicates that the paddle needs to be shifted towards the left. The accompanying image provides visual context, showing the current state of the game with the paddle positioned in the center. The task involves understanding the instruction and taking appropriate action in the game environment, which would likely involve pressing the corresponding control to move the paddle leftward.

This sample demonstrates how the Alpaca dataset combines textual instructions with visual information to create a rich training environment for instruction-based learning tasks.

5. What are the pertaining data used by LLaMA? How many tokens are in the entire training set for training LLaMA?
Ans: The pretraining data is a mixture from several sources as follows:
CommonCrawl - 67.0%

C4 - 15.0%
Github - 4.5%
Wikipedia - 4.5%
Books - 4.5%
ArXiv - 2.5%
StackExchange - 2.0%

Overall, the entire training dataset contains roughly 1.4T tokens after tokenization.

6. Before pre-training, what is the procedure for data preprocessing?
Ans:
English CommonCrawl [67%]: They preprocess five CommonCrawl dumps, ranging from 2017 to 2020, with the CCNet pipeline (Wenzek et al., 2020). This process deduplicates the data at the line level, performs language identification with a fastText linear classifier to remove non-English pages and filters low quality content with an ngram language model. In addition, they trained a linear model to classify pages used as references in Wikipedia v.s. randomly sampled pages, and discarded pages not classified as references.

C4 [15%]: The preprocessing of C4 also contains deduplication and language identification steps: the main difference with CCNet is the quality filtering, which mostly relies on heuristics such as presence of punctuation marks or the number of words and sentences in a webpage.

Github [4.5%]: They use the public GitHub dataset available on Google BigQuery. They only kept projects that are distributed under the Apache, BSD and MIT licenses. Additionally, they filtered low quality files with heuristics based on the line length or proportion of alphanumeric characters, and removed boilerplate, such as headers, with regular expressions. Finally, they deduplicate the resulting dataset at the file level, with exact matches.

Wikipedia [4.5%]: The authors process the data to remove hyperlinks, comments and other formatting boilerplate.

Gutenberg and Books3 [4.5%]: The authors include two book corpora in the training dataset: the Gutenberg Project, which contains books that are in the public domain, and the Books3 section of ThePile (Gao et al., 2020), a publicly available dataset for training large language models. They perform deduplication at the book level, removing books with more than 90% content overlap.

ArXiv [2.5%]. The authors process arXiv Latex files to add scientific data to the dataset. Following Lewkowycz et al. (2022), they removed everything before the first section, as well as the bibliography. They also removed the comments from the .tex files, and inline-expanded definitions and macros written by users to increase consistency across papers.

Stack Exchange [2%]: The authors kept the data from the 28 largest websites, removed the HTML tags from text and sorted the answers by score (from highest to lowest)

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

Tokenization refers to the process of converting a sequence of text into smaller parts, known as tokens. These tokens can be as small as characters or as long as words. This process helps machines understand human language. There are different types of tokenization like Word Tokenization, Character Tokenization and Subword Tokenization.

Few tokenization algorithms are:

1 - Whitespace Tokenization
2 - Rule-Based Tokenization
3 - Regular Expression Tokenization
4 - Subword Tokenization
5 - Treebank Tokenization

Tokenization method used by LLaMA is - Byte Pair Encoding (BPE) algorithm

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

Ans: The main three types of transformer architectures are:
1. Encoder-only Transformers: BERT, RoBERTa
2. Decoder-only Transformers: GPT, GPT-2, GPT-3
3. Encoder-Decoder Transformers: BART (Bidirectional and Auto-Regressive Transformers), T5 (Text-to-Text Transfer Transformer)

9. Why are LLMs mainly decoder-only architecture?
Ans: LLMs are mainly decoder-only architecture because they're designed for autoregressive generation, predicting tokens based on preceding context, which decoder architectures naturally support. Additionally, decoder-only models prevent "cheating" during training by masking future tokens.

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

Position embedding is a technique used in NLP to embed the position of each word in a sentence along with its meaning.Position embedding adds a position vector to each word embedding, which represents the position of the word in the sentence.The position vector is generated using a mathematical function called a positional encoding function. The positional encoding function takes two inputs: the position of the word in the sentence and the dimension

of the embedding. The output of the positional encoding function is a fixed-size vector that represents the position of the word in the sentence.

Algorithms:

Sinusoidal Position Embeddings
Learned Position Embeddings
Relative Position Embeddings
Rotary Position Embeddings (RoPE)
ALiBi (Attention with Linear Biases)

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

Language modeling is a foundational task that involves the process of predicting the likelihood of a sequence of words occurring in a language.A language model (LM) estimates the probability distribution of various linguistic units, such as words, phrases, or sentences, enabling it to predict the next word or sequence of words in a sentence.

Casual Language Modelling	Masked Language Modeling
Predict the next word in a sequence given the previous words. It's a generative task where the model learns to produce text.	Random tokens in a sentence are masked or hidden, and the model is trained to predict these masked tokens based on the context provided by the non-masked tokens
The model only has access to past and current information but not future context when making predictions	The model considers both past and future context within a sentence to make predictions
Example: GPT	Example: BERT

12. How to generate text from LLMs [5]? Explain different decoding strategies.

Ans:

Generating text from LLMs involves the process of selecting the next word in the sequence based on the model's prediction model. There are multiple decoding strategies that can be used to guide the generation process, each affecting the coherence and relevance of the generated text in different ways.
Different decoding strategies:

1 - Greedy Decoding

In this strategy, the prediction model selects the token which has the highest probability as its output at each step (till the stop condition is met).

Advantage - It is fast and computationally efficient.
Disadvantage - It produces repetitive/generic text.

2 - Beam Search

This strategy builds on the greedy decoding strategy by covering the top N most likely token at every step, keeping a fixed number of these sequences as candidates. It finally selects the overall best sequence based on cumulative probabilities.

Advantage - Coherent and less repetitive text when compared with greedy decoding.
Disadvantage - Computationally intensive

3 - Top-K Sampling

This strategy introduces randomness by limiting the pool of next-token candidates to the top-K most likely tokens and then sampling from this depending on the probabilities.

Advantage - Generates more diverse text by introducing variability in the selection process
Disadvantage - May lead to less coherent outputs if the value of K is set very high

4 - Top-p (nucleus) Sampling

Rather than selecting a fixed number of top tokens, this strategy chooses from the smallest set of tokens whose cumulative probability exceeds the threshold p.

Advantage - Balances coherence and diversity better than top-K
Disadvantage - Choice of p (too high or too low) affects the quality of text generated.

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

Ans:

Instruction tuning involves the process of fine tuning a pre-trained model on a dataset of instructions/prompts paired with the appropriate responses. The aim is to improve the ability of the model to understand and execute a wide variety of text-based instructions accurately.

Importance of Instruction Tuning:

1 - Significantly improves the ability of the model to comprehend and execute textual instructions

2 - Perform tasks that the model was not explicitly trained for

3 - Model can understand and execute tasks it has seen few or no examples of during training, solely based on the instructions given.

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

Ans:

Alignment tuning is aimed at aligning the model's output with human values, ethical standards, intentions and preferences. This aim can be achieved by fine tuning the model on a curated dataset where the desired outputs are based on feedback, preferences, or corrections provided by humans.

Importance of alignment tuning:

1 - works towards model generating outputs that are in line with human values, ethics and societal norms.

2 - works towards model generating outputs that comply with the evolving regulatory standards, avoiding legal and social ramifications.

3 - works towards model generating outputs that are appropriately sensitive depending upon the user group.

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

Ans:

Parameter Efficient fine-tuning is a technique that is used to improve the performance of pre-trained language models on specific tasks. It involves reusing the parameters of the pre-trained model and then fine tuning it on the specific smaller dataset which helps in saving **computational resources** and **time** as compared to training the entire model from scratch.

In PEFT, there are 2 notable strategies named Prefix tuning and Prompt Tuning

Prefix Tuning	Promt Tuning
More parameters updated (prefixes for each layer's input)	Fewer parameters updated (only the input prompt embeddings)
Provides better performance since more number of characters can capture the nuances better	Provides a lower performance as compared to prefix tuning due to limited parameter updates
Lower computational efficiency (due to more parameter updates)	Higher computational efficiency

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

Ans:

Prompt engineering is the process of designing the inputs to the models so as to get optimal/desired outputs.

Importance:

1 - Well designed inputs can significantly improve the model performance.
2 - Using prompt engineering, a single model can be applied to wide variety of tasks without task-specific training
3 - It also promotes cost efficiency since we can leverage the pre-trained models for new tasks without the computational and financial costs of retraining or fine-tuning.

Zero Shot Demonstration:

In this, the model is provided with a task along with a prompt without any specific details about performing it. The model relies on its pre-trained knowledge to generate an appropriate/optimal response. This approach is used to test the model's capacity for generalization - its ability to apply the pre-trained knowledge for performing novel tasks.

Few Shot Demonstration:

In this, the model is provided with some examples of the task within the prompt itself to demonstrate how to perform it. This helps the model in understanding the requirements well and the desired format of the response. This helps in significantly improving the model's performance on specific tasks.

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

Ans:

In-context learning	Chain-of-thought prompting
Focuses on the model's ability to learn from the provided context during the inference stage	Focuses around the user's approach to crafting interconnected prompts/inputs to guide the model through a conversation
Useful for broad variety of tasks due to its ability to adapt on the fly	Useful for tasks that require a detailed or coherent response from the model

18. What are the three basic types of ability evaluation for LLMs? What is perplexity [6]? What is hallucination?

Ans:

3 basic types of ability evaluation in LLMs are:

Linguistic Ability - It evaluates the ability of the model to generate the natural language which includes the grammar, syntax and semantics.

Knowledge and Reasoning - Assesses the ability of the model to store, recall, and apply factual knowledge and its capacity for logical reasoning, problem-solving, and understanding implicit information

Task-Specific Performance - Measures how well a model performs on specialized tasks that may require understanding specific domains.

What is perplexity -

It is the measurement that is used to evaluate the performance of the language models. It provides a quantitative way to assess a model's linguistic capabilities.
In terms of LLMs, it evaluates how well a model predicts the next token in the sequence. A low perplexity score indicates a better model performance in prediction.

What is hallucination -

It refers to the instances where a language model generates incorrect information but presents it as factual information. Hallucination is a big challenge in development and deployment of LLMs

specially in applications that need a high level of accuracy due to its criticality. Addressing hallucination in LLM outputs involves both improving the models' training and implementation methodologies and developing robust evaluation and filtering methods to detect and correct hallucinated content.

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

Ans:

Human alignment refers to the process of ensuring that the AI models act in a way such that they are aligned with human values, ethics and goals.

Importance:

1 - Human alignment ensures that AI models operate within ethical boundaries set by human society
2 - It helps in building trust among the users thus promoting adoption of these models
3 -It helps prevent unwanted or surprising results from outputs the model produces for tasks it wasn't specifically programmed for.

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

Ans:

Some comprehensive evaluation benchmarks:

1 - GLUE (General Language Understanding Evaluation)
2 - SuperGLUE
3 - LAMBADA (LAnguage Modeling Broadened to Account for Discourse Aspects)
4 - BIG-bench (Beyond the Imitation Game benchmark)
5 - HellaSwag

OPTIMIZATION METHODS:

3.1 Gradient Accumulation

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

Ans: $\frac{\delta L}{\delta w} = \left(\frac{1}{32} \sum_{i=1}^{32} \frac{\delta Lj}{\delta w} \right) \quad \frac{\delta L}{\delta w} = \frac{1}{4} \left(\frac{1}{8} \sum_{i=1}^{8} \frac{\delta Li}{\delta w} + \frac{1}{8} \sum_{j=1}^{8} \frac{\delta Lj}{\delta w} + \frac{1}{8} \sum_{k=1}^{8} \frac{\delta Lk}{\delta w} + \frac{1}{8} \sum_{l=1}^{8} \frac{\delta Ll}{\delta w} \right)$

The above equation shows a single gradient update step for the entire batch of 32 samples and accumulating gradient for 4 mini-batches each containing 8 samples. Dividing gradient of each mini-batch by 4 and then accumulating 4 mini-batches together yields exact same results.

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

Ans: Batch normalization statistics like mean and variance may not be consistent across mini-batches. As a result, gradients of the mini-batches will be calculated with different batch statistics. Thus, gradient accumulation may not lead identical results.

3.2 Gradient Checkpointing

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

Ans: The peak memory is dominated by activations that are stored for gradient computation during the backward pass. Model inference does not need gradient computation and therefore does not need to store activations, thereby alleviating the memory blow-up problem.

Q - What are the memory requirement and forward computation steps for the two strategies in big O notation? Check this medium post [7] for more illustrations.

Ans: Strategy 1:

- Memory Requirement: O(1)
- Compute Requirement: $O(n^2)$

Strategy 2:

- Memory Requirement: $O(\sqrt{n})$
- Compute Requirement: O(n)

3.3 Low-Rank Adaptation (LoRA)

1. What is matrix rank?

Ans:
Matrix rank is the maximum number of linearly independent column vectors or row vectors in the matrix—both counts will always be the same for any given matrix. If a matrix has full rank, it means that none of its rows or columns can be constructed by a linear combination of other rows or columns.

2. What are three decomposed matrices by SVD?

Ans:

$W = U\Sigma V^T$

Singular Value Decomposition (SVD) decomposes a matrix into three other matrices:

U: n x n orthogonal matrix whose columns are the left singular vectors of W

Σ: diagonal matrix (n x n) diagonal matrix with singular values on the diagonal such that $\sigma1>=\sigma2>=\sigma3….\sigma n>=0$

V^T:is the transpose of an n×n orthogonal matrix whose columns are the right singular vectors of W.

3. U and V are orthogonal matrices. Why does it imply UUT =I,VVT =I?

Ans:

For any orthogonal matrix A, the value of $A*A^T$ is an identity matrix I.

Since A is orthogonal, its columns are orthonormal. Each element of the matrix $A*A^T$ is a dot product of columns of A with rows of A^T.

For diagonal elements, since it would be the product of a column vector with itself , their dot product will be 1.

For non diagonal elements, since it would be the product of two distinct column vectors, their dot product will be 0.

Thus we will have an identity matrix.

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

Ans:

If a matrix has full rank, it means that all its rows and columns are linearly dependent, thus its rank is 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)?

Ans:

The reconstructed image will be able to retain the most significant and prominent features of the original image. These top singular values correspond to the most prominent patterns in the image. But since we are setting the rest of the singular values to 0, the reconstructed image will be missing the finer details thus resulting in lower or blurry resolution.

As we increase the value of k, the reconstructed image will progressively retain more details resulting in a higher resolution.

6. If a matrix W ∈ Rn×n is low rank, what does its singular matrix look like?

Ans:

If W has a low rank, it means that there are fewer linearly independent rows and columns.

Let us say r is the rank of the matrix W,

For a low rank matrix, the singular matrix Σ would have first r values of the its leading diagonal to be non zero singular values and the remaining n-r elements would be 0.

It implies that first r columns of U and first r rows of V^T contribute to matrix W when we take the product $U\Sigma V^T$

7. If the top-k singular values of a matrix W ∈ Rn×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 ≈ AB)?

Ans:

Yes we can represent W by two low rank matrices A and B.

We can write A as $U_k * (S_k)^{½}$, where U_k is n x k matrix (first k columns of U) and $(S_k)^{½}$ is a k x k diagonal matrix with top k singular values.

We can write B as $V^T_k * (S_k)^{½}$, where U_k is k x n matrix (first k rows of V^T) and $(S_k)^{½}$ is a k x k diagonal matrix with top k singular values.

Therefore W could be approximated to $W \approx U_k * (S_k) * V^T_k = AB$

Yes these two matrices can be a good approximation if multiplied together because we are able to effectively capture the essence of W in fewer dimensions. When we multiply A and B, the resulting matrix AB would be a rank - k matrix which approximates W, keeping its most prominent and significant characteristics while ignoring/discarding the less prominent ones.

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.

Ans:

Truncated SVD may not yield a good approximation in several situations, particularly when the singular values that are set to zero still carry important information about the matrix.

Situations where it might fail:

1 - When the inherent rank of the data is high, meaning that the data cannot be well-represented in a reduced-dimensional space without substantial information loss, truncated SVD will not be effective.

2 - If the dataset has a lot of noise, the singular values may reflect this noise, and it may not be clear which singular values correspond to noise and which to actual data.

3 - If the singular values are uniformly large or decay very slowly, it means that each singular value carries about the same amount of information. Truncating any of them could lead to significant loss of information.

Q - 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?

Ans:

Though intuitively it seems that adding more parameters to the model would increase the cost of forward pass and inference latency, it is not true in the case of LORA.

Forward Pass Costs:

LoRa introduces two low ranked matrices A and B which are quite smaller in size as compared to the weight matrix in the original model.
During the forward pass in the training phase, only the matrices A and B are updated (and the original weights are frozen) therefore significantly reducing the memory usage and number of

computations as compared to the traditional approach of fine tuning (updating the entire weight matrix). This ensures that the additional computational overhead during training is kept minimal.

Inference Latency:

The process of inference involves using the sum of the original pre-trained weights (W_0) and the low rank updates (ΔW). Since this sum is pre-computed after the training phase is completed, just before the inference phase, the model with LoRA can operate with the same inference speed as a standard model that has been fine-tuned, avoiding any additional inference latency.

3.4 Mixed Precision Training

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

Ans:

Need for master copy of weights:

1 - The FP32 master copy of weights maintains a high precision version of the weights of the model. Even though the gradients are computed in FP16 for efficiency, the values of these gradients can become too small to be represented in the FP16 format. These small valued gradients would become zero in the optimizer when multiplied with the learning rate and adversely affect the model accuracy. By using an FP32 master copy, these gradients can be scaled up and applied ensuring that every small update is reflected.

2 - Large weight to update ratio: If the weight is more than (or equal to) 2048 times the update, the shifting required for alignment can push the mantissa bits of the update entirely out of the representable range of FP16 (10 bits mantissa). Thus the update value would become essentially 0, resulting in no update to the weight.

Need for Scaling up the loss:

To solve the issue of the gradients becoming too small to be represented in FP16, the papers suggests to Scale up the loss - This also implies that all the gradients computed during the process of backpropagation would also be scaled by the same factor. This would ensure that the gradients remain in the representable range of FP16. After backpropagation, before the weight update, the gradients are scaled down by the same factor to maintain the update magnitudes as in FP32 training.

Q - Since activations are also stored in half- precision format, the overall memory consumption for training deep neural networks is roughly halved. ” Do you think this statement still holds for LLM fine-tuning?

Ans:

I think the statement does hold true for LLM fine-tuning as well.

LLMs due to their large number of parameters and layers, generate a huge number of activations. By storing the activations in a half precision format, fine tuning of LLMs can be made quite memory efficient thus allowing the use of large batch sizes or training larger models within the same memory constraints. This efficiency is pivotal for fine-tuning processes where the ability to process large amounts of data efficiently can directly impact the speed and effectiveness of the fine-tuning.