LLM

Q1. What is tokenization, and why is it important in LLMs?

This step is crucial because LLMs do not understand raw text directly. Instead, they process sequences of numbers that represent these tokens.

Q2. What is LoRA and QLoRA?

LoRA and QLoRA are techniques designed to optimize the finetuning of Large Language Models (LLMs), focusing on reducing memory usage and enhancing efficiency without compromising performance in Natural Language Processing (NLP) tasks.

LoRA (Low-Rank Adaptation)

LoRA is a parameter-efficient fine-tuning method that introduces new trainable parameters to modify a model's behavior without increasing its overall size. It works by adding low-rank matrix adaptations to the model's existing layers, allowing for significant performance improvements while keeping resource consumption in check.

QLoRA (Quantized LoRA)

QLoRA builds on LoRA by incorporating quantization to further optimize memory usage. It uses techniques such as 4-bit Normal Float, Double Quantization, and Paged Optimizers to compress the model's parameters and improve computational efficiency. By reducing the precision of model weights (e.g., from 16-bit to 4-bit) while retaining most of the model's accuracy.

Q3. What is beam search, and how does it differ from greedy decoding?

Beam search is a search algorithm used during text generation to find the most likely sequence of words. Instead of choosing the single highest-probability word at each step (as greedy decoding does), beam search explores multiple possible sequences in parallel, maintaining a set of the top k candidates (beams). It balances between finding high-probability sequences and exploring alternative paths. This leads to more coherent and contextually appropriate outputs, especially in long-form text generation tasks.

Q4. Explain the concept of temperature in LLM text generation.

Temperature is a hyperparameter that controls the randomness of text generation by adjusting the probability distribution over possible next tokens. A low temperature (close to 0) makes the model highly deterministic, favoring the most probable tokens. Conversely, a high temperature (above 1) encourages more diversity by flattening the distribution, allowing less probable tokens to be selected.

Q5. What is masked language modeling, and how does it contribute to model pretraining?

- Masked language modeling (MLM) is a training objective where some tokens in the input are randomly masked, and the model is tasked with predicting them based on context. This forces the model to learn contextual relationships between words, enhancing its ability to understand language semantics. MLM is commonly used in models like BERT, which are pretrained using this objective to develop a deep understanding of language before fine-tuning on specific tasks.

**Q6. What are Sequence-to-Sequence Models?**

**Ans -** Sequence-to-Sequence (Seq2Seq) Models are a type of neural network architecture designed to transform one sequence of data into another sequence. These models are commonly used in tasks where the input and output have variable lengths, such as in machine translation, text summarization, and speech recognition.

**Q7. How do autoregressive models differ from masked models in LLM training?**

Autoregressive models, such as GPT, generate text one token at a time, with each token predicted based on the previously generated tokens. This sequential approach is ideal for tasks like text generation. Masked models, like BERT, predict randomly masked tokens within a sentence, leveraging both left and right context. Autoregressive models excel in generative tasks, while masked models are better suited for understanding and classification tasks.

**Q8. What role do embeddings play in LLMs, and how are they initialized?**

**Ans -** Embeddings are dense, continuous vector representations of tokens, capturing semantic and syntactic information. They map discrete tokens (words or subwords) into a high-dimensional space, making them suitable for input into neural networks. Embeddings are typically initialized randomly or with pretrained vectors like Word2Vec or GloVe. During training, these embeddings are finetuned to capture task-specific nuances, enhancing the model’s performance on various language tasks.

**Q9. What is next sentence prediction and how is useful in language modelling?**

**Ans -** Next Sentence Prediction (NSP) is a key technique used in language modeling, particularly in training large models like BERT (Bidirectional Encoder Representations from Transformers). NSP helps a model understand the relationship between two sentences, which is important for tasks like question answering, dialogue generation, and information retrieval.

**Q10. Explain the difference between top-k sampling and nucleus (top-p) sampling in LLMs.**

**Ans -** Top-k sampling restricts the model’s choices to the top k most probable tokens at each step, introducing controlled randomness. For example, setting k=10 means the model will only consider the 10 most likely tokens. Nucleus sampling, or top-p sampling, takes a more dynamic approach by selecting tokens whose cumulative probability exceeds a threshold p (e.g., 0.9). This allows for flexible candidate sets based on context, promoting both diversity and coherence in generated text.

**Q11. How does prompt engineering influence the output of LLMs?**

LLMs are highly sensitive to input phrasing, a well-designed prompt can significantly influence the quality and relevance of the response. For example, adding context or specific instructions within the prompt can improve accuracy in tasks like summarization or question-answering.

**Q12. How can catastrophic forgetting be mitigated in large language models (LLMs)?**

**Ans -** Catastrophic forgetting happens when an LLM forgetspreviously learned tasks while learning new ones, which limits itsversatility. To mitigate this, several strategies are used:

1. Rehearsal methods: retraining model on a mix of old and new data.
2. Elastic weight consolidation: assign importance to certain model weights.
3. Modular approach: progressive neural networks and optimized fixed expansion layers introduce new modules for new task thus allowing LLM to learn without overwriting existing layers.

**Q13. What is model distillation, and how is it applied to LLMs?**

**Ans -** Model distillation is a technique where a smaller, simpler model (student) is trained to replicate the behavior of a larger, more complex model (teacher). In the context of LLMs, the student model learns from the teacher’s soft predictions rather than hard labels, capturing nuanced knowledge. This approach reduces computational requirements and memory usage while maintaining similar performance, making it ideal for deploying LLMs on resource-constrained devices.

**Q14. How do LLMs handle out-of-vocabulary (OOV) words? Ans -** Out-of-vocabulary words refer to words that the model did not encounter during training. LLMs address this issue through subword tokenization techniques like Byte-Pair Encoding (BPE) and WordPiece. These methods break down OOV words into smaller, known subword units. For example, the word “unhappiness” might be tokenized as “un,” “happi,” and “ness.” This allows the model to understand and generate words it has never seen before by leveraging these subword components.

**Q15. How does the Transformer architecture overcome the challenges faced by traditional Sequence-to-Sequence models?**

**Parallelization:** Seq2Seq models process sequentially, slowing training. Transformers use self-attention to process tokens in parallel, speeding up both training and inference.

Long range dependencies: Transformers capture long range dependencies using self-attention, allowing the model on any part of the sequence, regardless of distance.

Positional Encoding: Since transformers process the entire sequence at once. Positional encoding is used to make sure that the model understands the tokens in order.

Context bottleneck: seq2seq uses single context vector limiting information flow. Transformer let the decoder to attend all the encoder outputs, improving context retention.

**Q16. What is overfitting in machine learning, and how can it be prevented?**

**Ans -** Overfitting occurs when a machine learning model performs well on the training data but poorly on unseen or test data. This typically happens because the model has learned not only the underlying patterns in the data but also the noise and outliers, making it overly complex and tailored to the training set. As a result, the model fails to generalize to new data.

**Regularization (L1, L2):** Adding a penalty to the loss function to discourage overly complex models. L1 (Lasso) can help in feature selection, while L2 (Ridge) smooths weights.

Dropout: In neural networks, dropout randomly deactivates a fraction of neurons during training, preventing the model from becoming overly reliant on specific nodes.

Data Augmentation: expanding the training dataset with slight variations.

Early Stopping: monitoring the performance of the model on validation data and stopping training when the validation loss stops decreasing.

Simpler Models: reducing the complexity of the model by decreasing the number of features, parameters or layers can help avoid overfitting.

**Q17. What are Generative and Discriminative models?**

**Generative models** learn the underlying data distribution and generate new samples from it. They model the joint probability distribution of inputs and outputs, aiming to maximize the likelihood of the observed data. A common example is a language model, which predicts the next word in a sequence based on previous words.

**Discriminative models** focus on learning a decision boundary between different classes in the input-output space. They model the conditional probability of outputs given inputs, aiming to accurately classify new examples. An example is a sentiment analysis model, which classifies text as positive, negative, or neutral based on its content. In short, generative models generate data, while discriminative models classify it.

**Q18. How is GPT-4 different from its predecessors like GPT-3 in terms of capabilities and applications?**

**Improved Understanding:** GPT-4 has roughly 1 trillion parameters, significantly more than GPT-3’s 175 billion parameters.

Multimodal capabilities, larger context window, batter accuracy and fine-tuning, language support.

**Q19. What are positional encodings in the context of large language models?**

**Ans -** Positional encodings are essential in Large Language Models (LLMs) to address the inability of transformer architectures to capture sequence order. Since transformers process tokens simultaneously through self-attention, they are unaware of token order.

**Q20. What is Multi-head attention?**

**Ans -** Multi-head attention is an enhancement of single-head attention, allowing a model to attend to information from different representation subspaces simultaneously, focusing on various positions in the data. Instead of using a single attention mechanism, Ulti-head attention projects the queries, keys, and values into multiple subspaces (denoted as h times) through distinct learned linear transformations.

**Q21. Derive the softmax function and explain its role in attention mechanisms.**

The softmax function transforms a vector of real numbers into a probability distribution.

A black and white math symbols

Description automatically generated

This ensures all output values lie between 0 and 1 and sum to 1,making them interpretable as probabilities

**Q22. How is the dot product used in self-attention, and what are its implications for computational efficiency?**

In self-attention, the dot product is used to calculate the similarity between query (Q) and key (K) vectors. The attention scores are computed as:

A black and white image of a mathematical equation

Description automatically generated

**Q23. Explain cross-entropy loss and why it is commonly used in language modeling.**

**Ans -** Cross-entropy loss measures the difference between the predicted probability distribution and the true distribution (one-hot encoding of the correct token). It is defined as:

A black and white math equation

Description automatically generated

**Q24. How do you compute the gradient of the loss function with respect to embeddings?**

**Ans -** To compute the gradient of the loss L with respect to an embedding vector E, you apply the chain rule:  
A black and white image of letters and lines

Description automatically generated with medium confidence

**Q25. What is the role of the Jacobian matrix in backpropagation through a transformer model?**

**Ans -** The Jacobian matrix represents the partial derivatives of a vector-valued function with respect to its inputs. In backpropagation, it captures how each element of the output vector changes with respect to each input. For transformer models, the Jacobian is essential in computing gradients for multi-dimensional outputs, ensuring that each parameter (including weights and embeddings) is updated correctly to minimize the loss function.