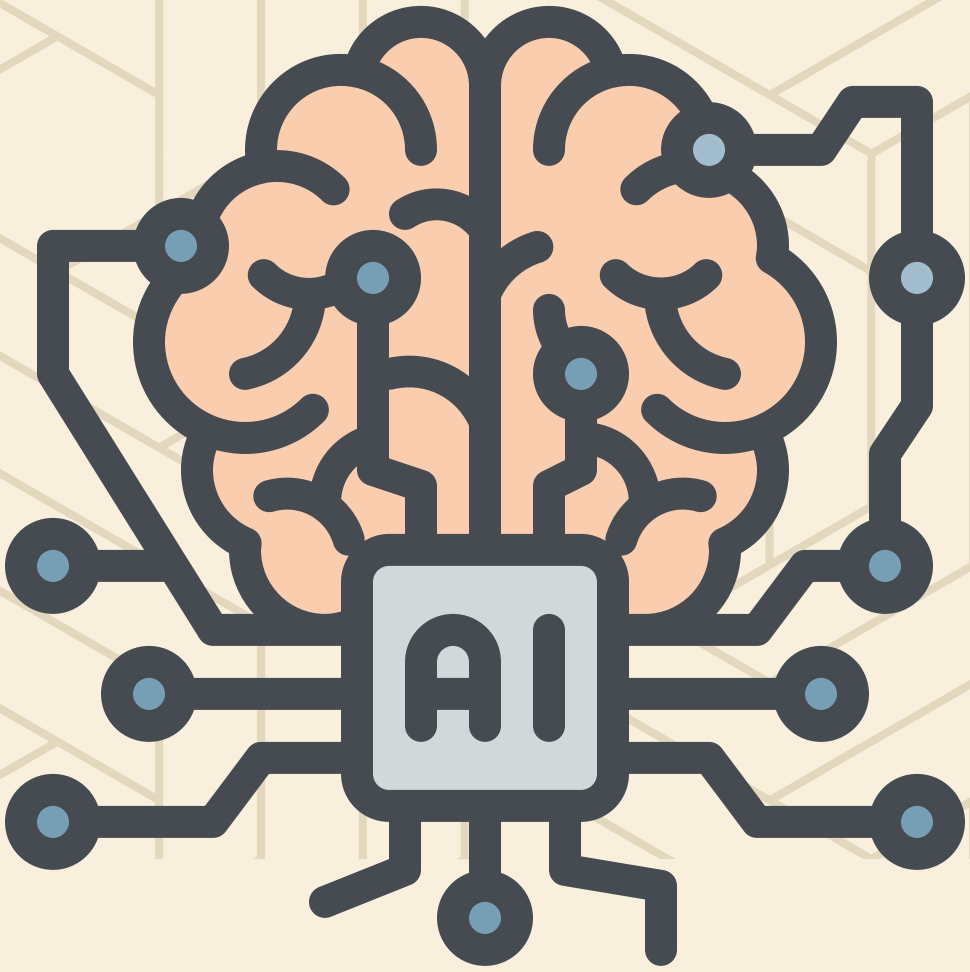
TOP 10 LLM

INTERVIEW QUESTIONS

Part II



Bhavishya Pandit

Part I

IN CASE YOU MISSED THE FIRST PART

Q1. What are Large Language Models?

Q2. How do you measure the performance of an LLM?

Q3. Explain the concept of “few-shot learning” in LLMs and its advantages.

Q4. You are working on an LLM, and it starts generating offensive or factually incorrect outputs. How would you diagnose and address this issue?

Q5. How is encoder different from the decoder?

Q6. What are the main differences between LLMs and traditional statistical language models?

Q7. What is a “context window”?

Q8. What is a hyperparameter?

Q9. Can you explain the concept of attention mechanisms in transformer models?

Q10. What are some common challenges associated with using LLMs?

For answers refer link in the comments

Q1. 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. Positional encodings provide the necessary information to help the model understand the sequence of words.

Mechanism:

- Additive Approach: Positional encodings are added to input word embeddings, merging static word representations with positional data.
- Sinusoidal Function: Many LLMs, such as the GPT series, use trigonometric functions to generate these positional encodings.

Formula:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

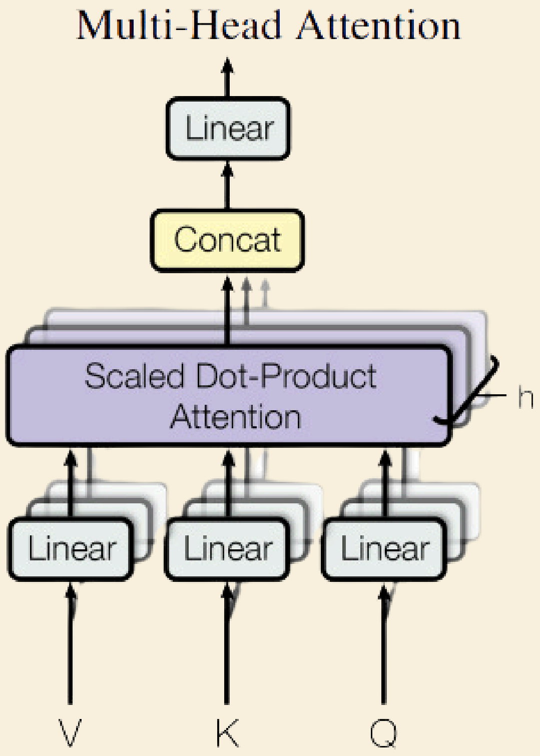
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$

Where:

- pos is the position in the sequence
- i is the dimension index (0 ≤ i < d_model/2)
- d_model is the dimensionality of the model

Q2. 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, multi-head attention projects the queries, keys, and values into multiple subspaces (denoted as h times) through distinct learned linear transformations.



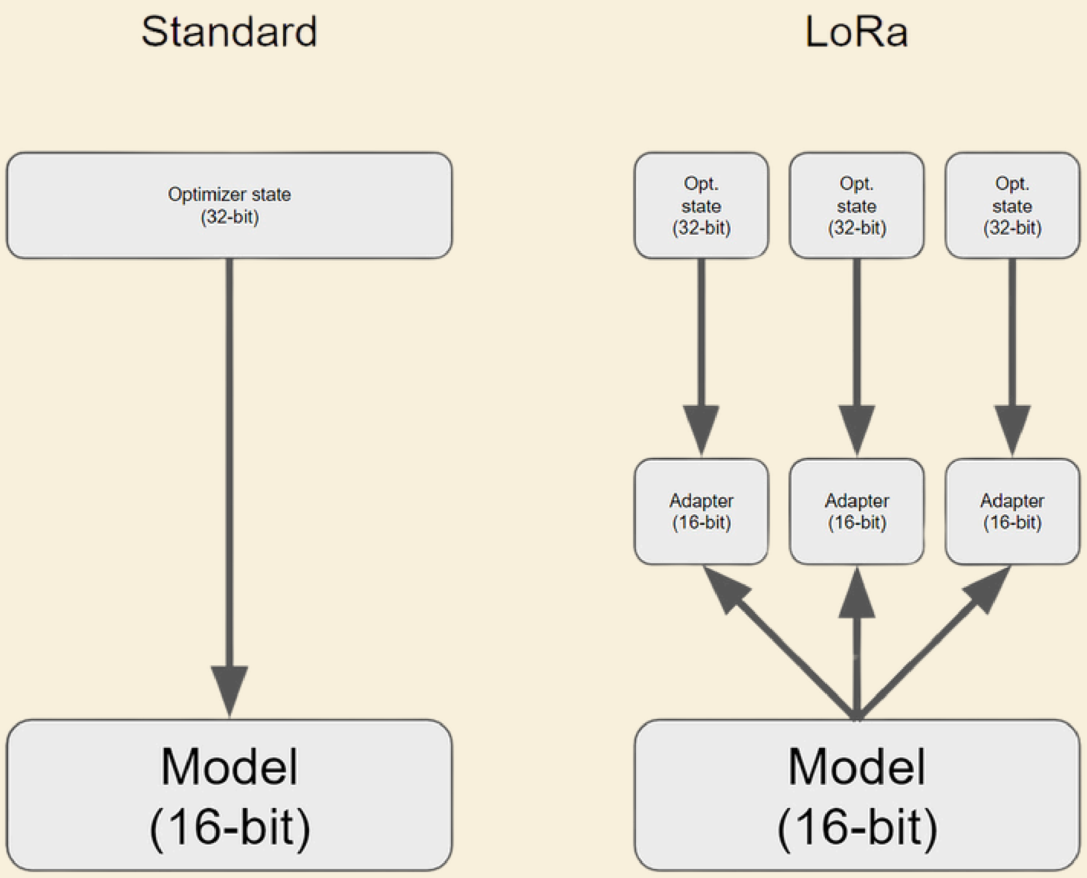
This process involves applying the attention function in parallel to each of these projected versions of the queries, keys, and values, which generates multiple output vectors. These outputs are then combined to produce the final dv-dimensional result. This approach improves the model's ability to capture more complex patterns and relationships in the data.

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{Attention}(Q_1, K_1, V_1), \ldots, \mathrm{Attention}(Q_h, K_h, V_h))W^O$$

LLM

Q3. What is LoRA and QLoRA?

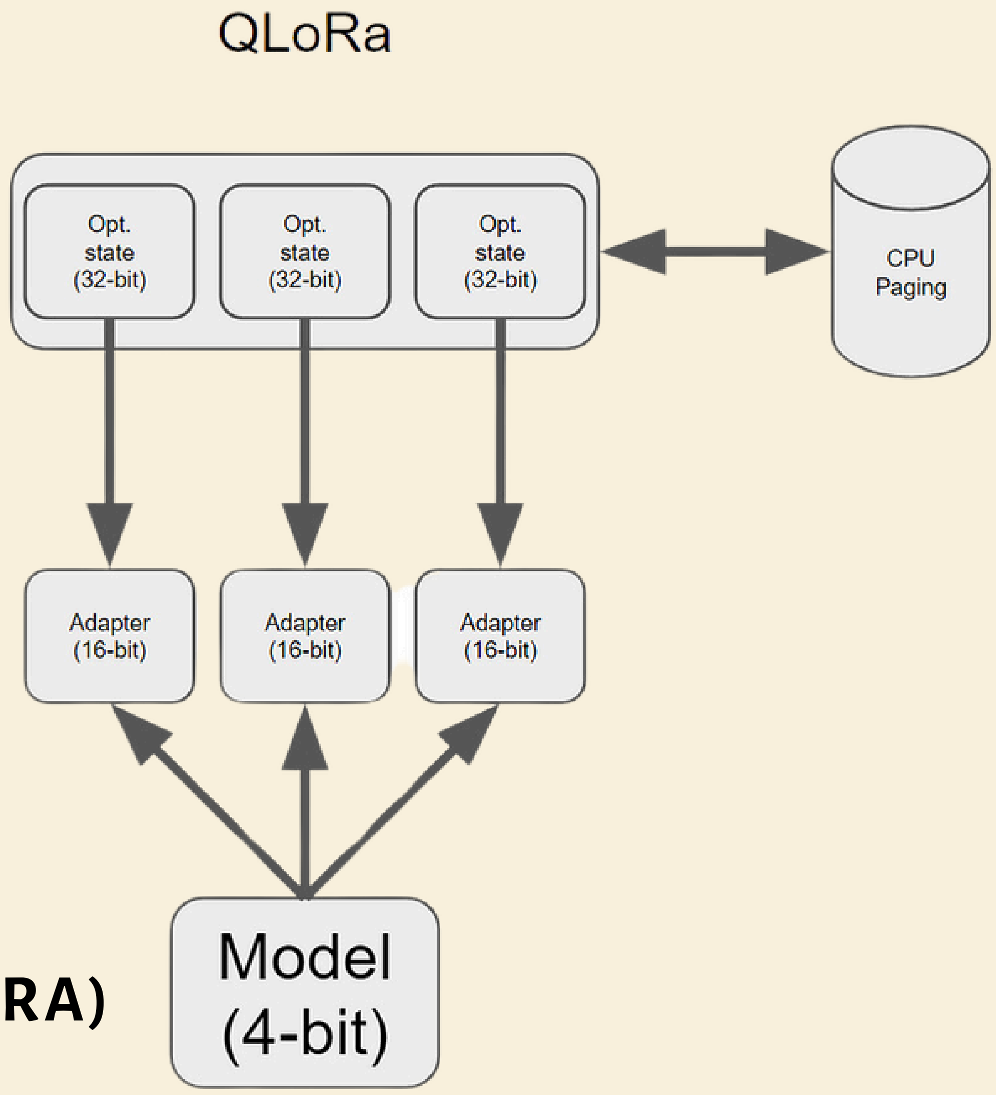
Ans - LoRA and QLoRA are techniques designed to optimize the fine-tuning of Large Language Models (LLMs), focusing on reducing memory usage and enhancing efficiency without compromising performance in Natural Language Processing (NLP) tasks.



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. By doing so, LoRA maintains the original parameter count, reducing the memory overhead typically associated with training large models. 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.

This makes it ideal for environments where computational resources are limited, yet high model accuracy is still required.

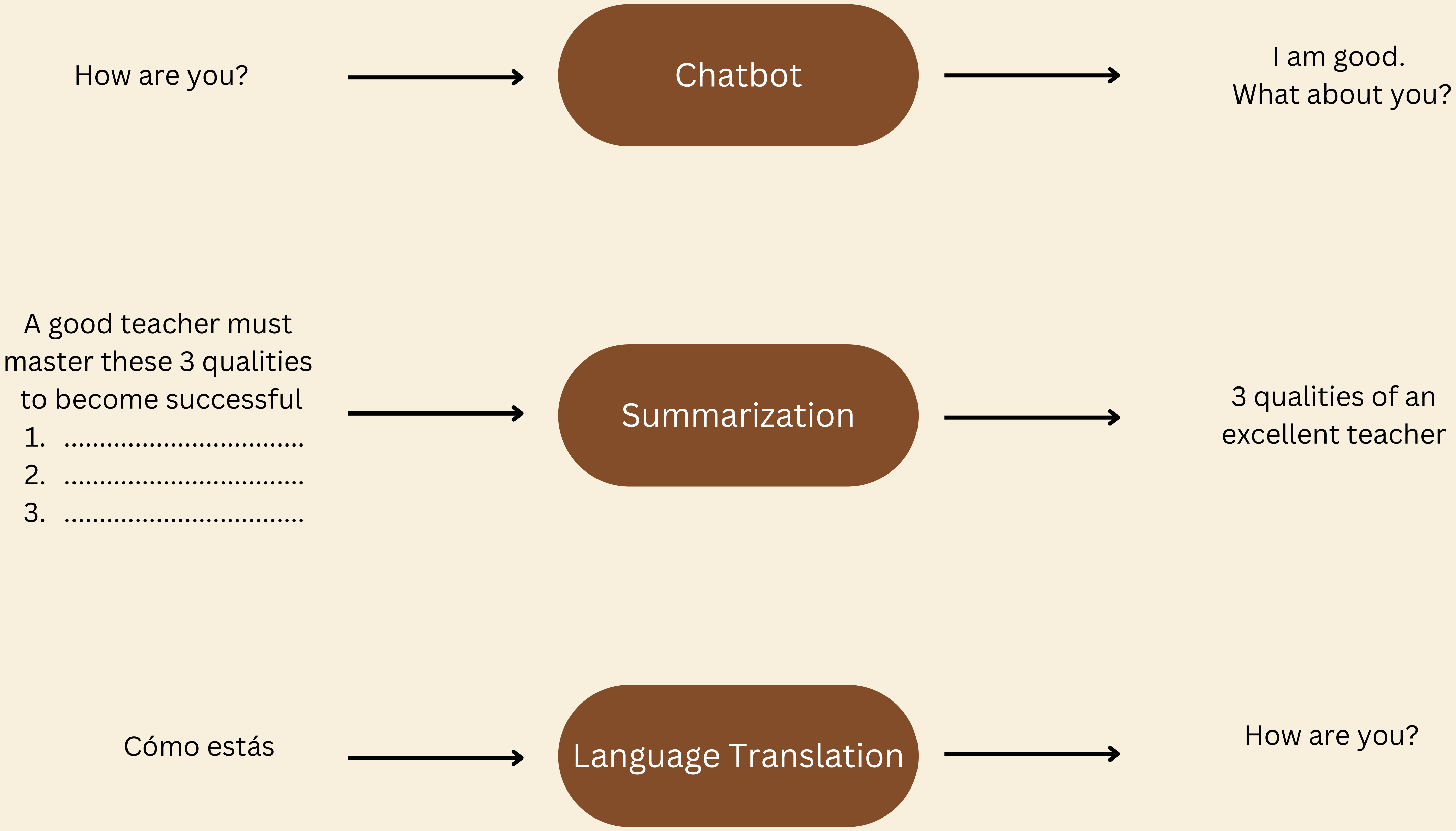


QLoRA (Quantized LoRA)

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, QLoRA allows for the fine-tuning of LLMs with minimal memory footprint. This method is particularly useful when scaling large models, as it maintains performance levels comparable to full-precision models while significantly reducing resource consumption.

Q4. 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.



How are you? → Chatbot → I am good. What about you?

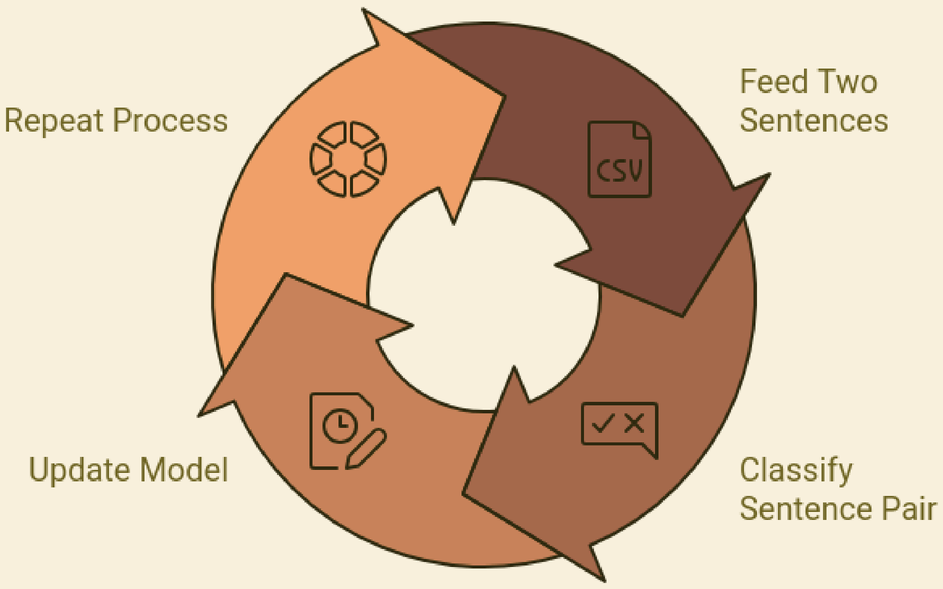
A good teacher must master these 3 qualities to become successful
1. ....
2. ....
3. ....
→ Summarization → 3 qualities of an excellent teacher

Cómo estás → Language Translation → How are you?

Q5. 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.

Next Sentence Prediction Training Cycle



During pre-training, the model is fed two sentences:

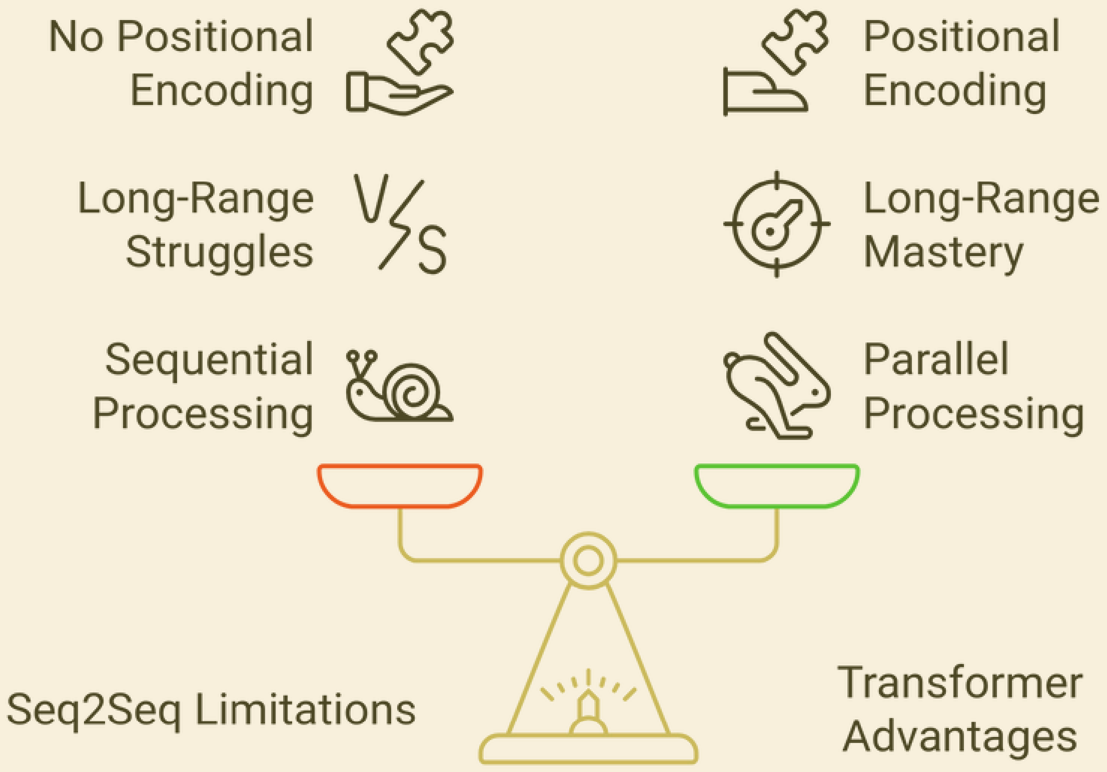
- 50% of the time, the second sentence is the actual next sentence in the document (positive pairs).

- 50% of the time, the second sentence is a random sentence from the corpus (negative pairs). The model is trained to classify whether the second sentence is the correct next sentence or not. This binary classification task is used alongside a masked language modeling task to improve the model's overall language understanding.

Q6. How does the Transformer architecture overcome the challenges faced by traditional Sequence-to-Sequence models?

Ans - The Transformer architecture overcomes key limitations of traditional Seq2Seq models in several ways:

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



Transformers outperform Seq2Seq models in key aspects.

- **Long-Range Dependencies:** Seq2Seq models struggle with long-range dependencies. Transformers capture these effectively with self-attention, allowing the model to focus on any part of the sequence, regardless of distance.

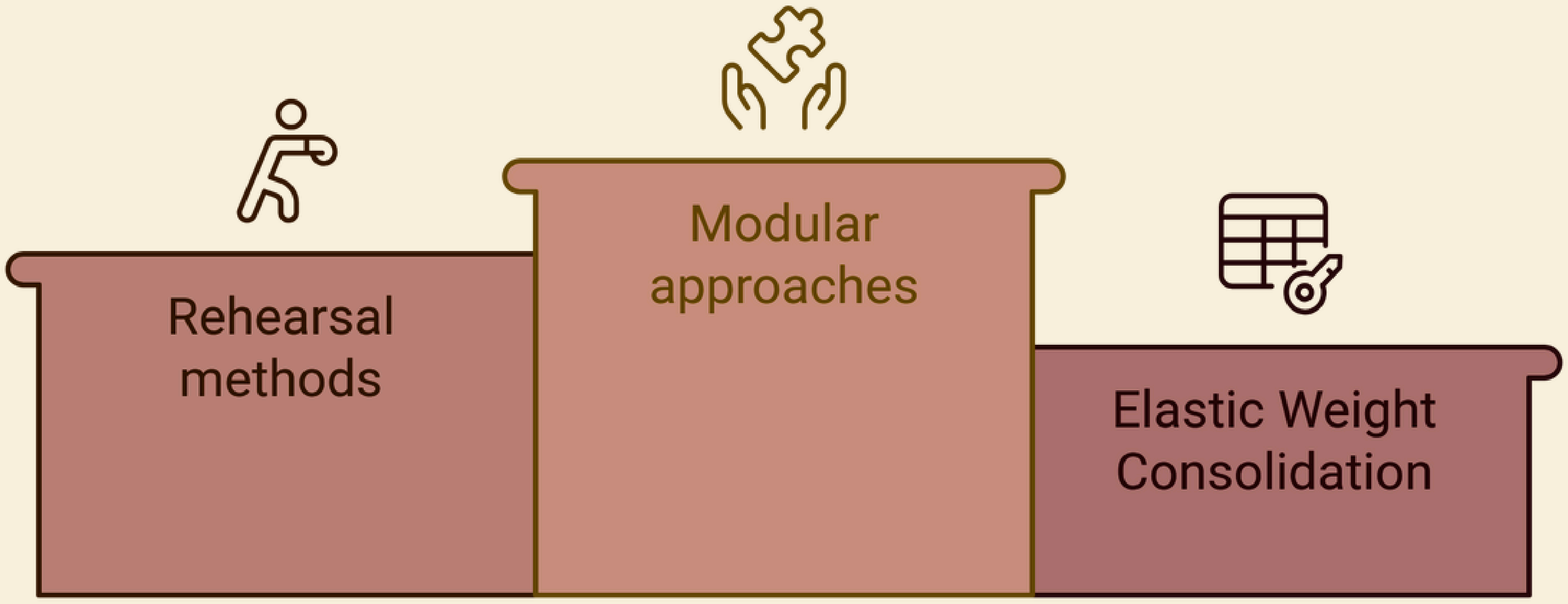
- **Positional Encoding:** Since Transformers process the entire sequence at once, positional encoding is used to ensure the model understands token order.

- **Efficiency and Scalability:** Seq2Seq models are slower to scale due to sequential processing. Transformers, with their parallelism, scale better for large datasets and long sequences.

- **Context Bottleneck:** Seq2Seq uses a single context vector, limiting information flow. Transformers let the decoder attend to all encoder outputs, improving context retention.

Q7. How can catastrophic forgetting be mitigated in large language models (LLMs)?

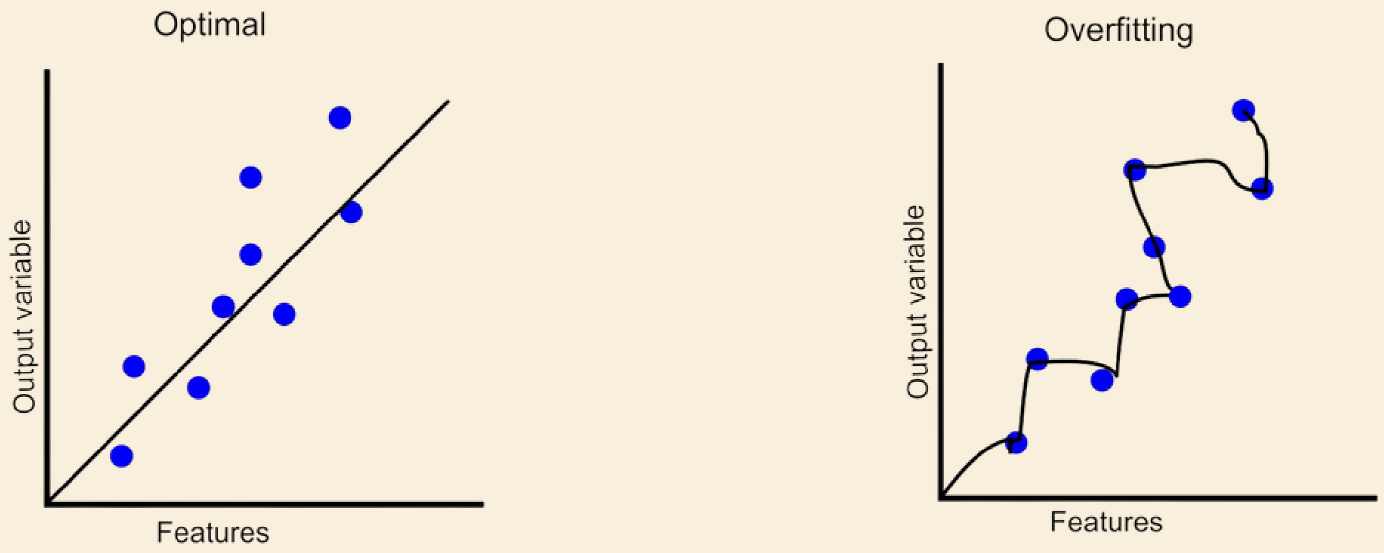
Ans - Catastrophic forgetting happens when an LLM forgets previously learned tasks while learning new ones, which limits its versatility. To mitigate this, several strategies are used:



1. **Rehearsal methods:** These involve retraining the model on a mix of old and new data, helping it retain knowledge of previous tasks.
2. **Elastic Weight Consolidation (EWC):** This method assigns importance to certain model weights, protecting critical knowledge while learning new tasks.
3. **Modular approaches:** Techniques like progressive neural networks (ProgNet) and optimized fixed expansion layers (OFELs) introduce new modules for new tasks, allowing the LLM to learn without overwriting prior knowledge.

Q8. 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.



Techniques to Overcome Overfitting:

- **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, such as flipping or rotating images, to make the model more robust.

- **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.

Q9. What are Generative and Discriminative models?

Ans - In NLP, generative and discriminative models are two key types of models used for various tasks.

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.

Generative Models

Generate new data

Discriminative Models

Classify existing data

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.

Q10. How is GPT-4 different from its predecessors like GPT-3 in terms of capabilities and applications?

Ans - GPT-4 introduces several advancements over its predecessor, GPT-3, in terms of both capabilities and applications:

better accuracy

larger context window

multimodal

Trillions of parameters

language support

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

- **Multimodal Capabilities:** GPT-4 processes both text and images, a major leap over GPT-3, which is text-only.

- **Larger Context Window:** GPT-4 can handle inputs with a much larger context window of up to 25,000 tokens, while GPT-3 maxes out at 4,096 tokens.

- **Better Accuracy and Fine-Tuning:** GPT-4 has been fine-tuned to be more factually accurate, reducing the likelihood of producing false or harmful information.

- **Language Support:** GPT-4 has improved multilingual performance, supporting up to 26 languages with higher accuracy compared to GPT-3's performance in non-English languages.

Follow for more AI/ML posts

SAVE

LIKE

SHARE