GenAI
Interview Prep

By Akash

Index

1. Generative AI Basics

1. **Q: What is Generative AI and how does it differ from other types of AI?**

   A: Generative AI creates new data samples similar to the training data. Unlike discriminative models that classify input data into categories, generative models generate new instances resembling the input data, such as text and image synthesis.

2. **Q: What is Generative AI?**

   A: Generative AI refers to a class of artificial intelligence models that can generate new data or content based on the data they were trained on, such as text, images, and audio.

3. **Q: What are some common applications of Generative AI?**

   A: Applications include text generation, image creation, music composition, data augmentation, drug discovery, and medical imaging enhancement.

4. **Q: How does a Generative Adversarial Network (GAN) work?**

   A: A GAN consists of two neural networks: a generator that creates fake data, and a discriminator that tries to distinguish between real and fake data. They are trained simultaneously, where the generator improves at creating realistic data, and the discriminator gets better at detecting fakes.

5. **Q: Can you explain the concept of 'latent space' in Generative AI?**

   A: Latent space is a lower-dimensional representation where generative models like GANs encode high-dimensional data. Navigating this space allows the model to generate new, similar data.

2. Generative AI Models and Techniques

6. **Q: What is a Variational Autoencoder (VAE)?**

   A: A VAE is a generative model that encodes input data into a probabilistic latent space and decodes it back to the original space, incorporating a regularization term to ensure the latent space follows a known distribution.

7. **Q: What is the difference between a Generative Adversarial Network (GAN) and a Variational Autoencoder (VAE)?**

A: GANs have two networks (generator and discriminator) competing to produce realistic data, while VAEs encode data into a latent space and decode it back, focusing on generating data by learning the underlying distribution.

8. **Q: How does the Encoder-Decoder architecture work?**

   A: The Encoder-Decoder architecture consists of an encoder that compresses input data into a fixed-size context vector and a decoder that generates the output from this vector, commonly used in tasks like machine translation.

9. **Q: What are the key components of a generative model's architecture?**

   A: Key components include the encoder (for encoding input data into latent space), the decoder (for reconstructing data from latent space), and loss functions (for guiding training).

10. **Q: How does an autoencoder work, and what is its purpose in Generative AI?**

    A: An autoencoder compresses input data into a lower-dimensional latent space and reconstructs it. It is used for tasks like dimensionality reduction, anomaly detection, and as a component in generative models.

11. **Q: How does the training process of a GAN work?**

    A: GAN training involves a generator creating fake data samples and a discriminator evaluating them against real samples. The generator aims to produce more realistic samples, while the discriminator improves its ability to distinguish real from fake data.

12. **Q: What is the role of loss functions in training GANs?**

    A: Loss functions guide the training of both the generator and discriminator. The generator's loss measures its ability to fool the discriminator, while the discriminator's loss measures its ability to detect fake data. Balancing these losses is crucial for stable training.

13. **Q: How can Generative AI be used in data augmentation?**

    A: Generative AI creates synthetic data that resembles real data to augment training datasets, improving machine learning model performance by increasing data diversity.

14. **Q: What challenges are associated with training Generative AI models?**

    A: Challenges include mode collapse (limited data variation), training stability, and the need for large datasets and computational resources.

3. Discriminative AI

15. **Q: What is Discriminative AI and how does it differ from Generative AI?**

    A: Discriminative AI models classify input data into predefined categories, learning the boundary between classes. Unlike generative models, they do not generate new data but classify existing data.

16. **Q: Can you give an example of a discriminative model and its application?**

    A: An example is a Support Vector Machine (SVM), which classifies data by finding the hyperplane that best separates different classes. Applications include image classification, spam detection, and medical diagnosis.

17. **Q: How does a discriminative model learn from data?**

    A: Discriminative models optimize a loss function measuring the difference between predicted and actual labels, using techniques like gradient descent to improve accuracy.

18. **Q: What are the key differences in the training objectives of generative and discriminative models?**

    A: Generative models learn the underlying data distribution to generate new samples, while discriminative models learn the decision boundary between classes.

19. **Q: What are some advantages of using discriminative models over generative models?**

    A: Discriminative models require less computational resources, have simpler training processes, and often achieve higher accuracy in classification tasks.

4. Applications and Use Cases

20. **Q: How can generative and discriminative models complement each other in a machine learning pipeline?**

    A: Generative models can create additional training samples for data augmentation, improving discriminative model performance. They can also help understand data distribution, informing discriminative model design and training.

21. **Q: Can you discuss a scenario where both generative and discriminative models are used together?**

A: In semi-supervised learning, generative models generate synthetic data to augment small labeled datasets, enhancing discriminative model training and classification performance.

22. **Q: How does Generative AI improve natural language understanding?**

    A: Generative AI models like GPT-4 can generate coherent and contextually relevant text, enhancing chatbots, machine translation, text summarization, and sentiment analysis tools.

23. **Q: What is the significance of "zero-shot learning" in Generative AI?**

    A: Zero-shot learning allows models to make predictions for unseen classes during training by leveraging features learned from seen classes, enabling content creation and task solving without extensive labeled data for every category.

24. **Q: How can Generative AI be used for anomaly detection?**

    A: Generative AI models can learn normal data patterns and identify deviations, making them effective for detecting anomalies in various applications like fraud detection and industrial monitoring.

25. **Q: What is adversarial training in Generative AI?**

    A: Adversarial training involves training models to be robust against adversarial attacks by incorporating adversarial examples in the training process, enhancing model security and reliability.

26. **Q: What are some ethical concerns associated with Generative AI?**

    A: Ethical concerns include misuse in creating fake content, intellectual property issues, bias in generated content, and privacy impacts. Ensuring responsible use and developing AI content detection techniques are ongoing challenges.

5. Transformer Architecture

27. **Q: What is the Transformer architecture and why is it significant in Generative AI?**

    A: The Transformer architecture, introduced by Vaswani et al. in 2017, uses self-attention to handle sequential data, making it significant for efficiently processing long-range dependencies in tasks like text generation and translation.

28. **Q: What is the purpose of the self-attention mechanism in the Transformer model?**

A: Self-attention allows the Transformer model to weigh the importance of different words in a sentence relative to each other, improving context understanding and prediction accuracy.

29. **Q: Can you explain the self-attention mechanism in Transformers?**

    A: Self-attention allows the model to weigh the importance of different words in a sequence, computing attention scores for each word relative to others, crucial for understanding contextual relationships.

30. **Q: How does positional encoding work in Transformers?**

    A: Positional encoding adds information about word positions in sequences using sine and cosine functions of different frequencies, allowing the model to understand word order.

31. **Q: What are the main components of the Transformer architecture?**

    A: The main components are the encoder and decoder stacks, each consisting of multiple layers of self-attention, feed-forward neural networks, layer normalization, and residual connections.

6. Large Language Models (LLMs)

32. **Q: What are Large Language Models (LLMs)?**

    A: LLMs are deep learning models trained on vast amounts of text data to understand and generate human-like language, used in applications like chatbots, translation, and content creation.

33. **Q: How do LLMs handle context in text generation?**

    A: LLMs use self-attention mechanisms to capture and weigh contextual information from large text corpora, enabling them to generate coherent and contextually relevant text.

7. Embedding Models

34. **Q: What are embedding models and their role in NLP?**

A: Embedding models convert text into numerical vectors representing semantic meanings, facilitating tasks like text classification, sentiment analysis, and similarity detection.

35. **Q: How do word embeddings improve NLP tasks?**

    A: Word embeddings capture semantic relationships between words, improving model performance by enabling better understanding of context and word similarities.

8. Retrieval-Augmented Generation (RAG)

36. **Q: What is Retrieval-Augmented Generation (RAG)?**

    A: RAG combines retrieval-based and generative models, retrieving relevant documents from a knowledge base to enhance text generation, improving accuracy and relevance in tasks like question answering.

37. **Q: How does RAG improve text generation tasks?**

    A: RAG retrieves contextually relevant information from external sources, providing additional context and facts that enhance the quality and accuracy of generated text.

9. Fine-Tuning and Prompt Engineering

38. **Q: What is LLM fine-tuning?**

    A: LLM fine-tuning involves adapting pre-trained large language models to specific tasks or domains using smaller, task-specific datasets, improving model performance on specialized tasks.

39. **Q: How does fine-tuning differ from pre-training?**

    A: Pre-training involves training a model on a large, general corpus, while fine-tuning adapts the pre-trained model to specific tasks with additional training on targeted datasets.

40. **Q: What is prompt engineering in the context of LLMs?**

    A: Prompt engineering involves designing effective input prompts to guide large language models in generating desired outputs, crucial for improving model performance on specific tasks.

41. **Q: How does prompt engineering enhance LLM performance?**

A: By carefully crafting prompts, prompt engineering can influence the model to produce more accurate, relevant, and contextually appropriate responses, optimizing task outcomes.

42. **Q: What is one-shot prompting?**

    A: One-shot prompting involves providing a single example to the model to guide its response, useful in scenarios with limited data availability.

43. **Q: How does one-shot prompting benefit NLP tasks?**

    A: It enables the model to learn from minimal input, making it efficient for tasks where providing extensive training data is impractical.

44. **Q: What is few-shot prompting?**

    A: Few-shot prompting involves providing a few examples to guide the model's response, striking a balance between data efficiency and task performance.

45. **Q: When is few-shot prompting particularly useful?**

    A: It is useful in scenarios where moderate amounts of example data are available, enhancing model accuracy without extensive training.

46. **Q: What is zero-shot prompting?**

    A: Zero-shot prompting involves guiding the model to perform tasks without any task-specific examples, relying on the model's pre-trained knowledge.

47. **Q: How does zero-shot prompting work?**

    A: The model leverages its understanding of language and context from pre-training to generate responses to unseen tasks based on general knowledge.

48. **Q: What is chain of thought prompting?**

    A: Chain of thought prompting involves breaking down complex tasks into smaller, sequential steps, guiding the model through a logical progression.

49. **Q: Why is chain of thought prompting effective?**

    A: It helps the model handle complex tasks by simplifying problem-solving processes, improving accuracy and coherence in generated responses.

50. **Q: When should chain of thought prompting be used?**

A: It is beneficial for tasks requiring multi-step reasoning or detailed explanations, enhancing the model's ability to generate thorough and logical outputs.

51. **Q: What is hybrid prompting?**

    A: Hybrid prompting combines multiple prompting techniques to leverage their strengths, optimizing model performance across diverse tasks.

52. **Q: How does hybrid prompting enhance LLM capabilities?**

    A: By integrating different prompting strategies, hybrid prompting can address task-specific challenges more effectively, improving overall response quality.

53. **Q: What is ReAct prompting?**

    A: ReAct prompting involves incorporating reactive feedback into the prompting process, allowing the model to adapt and refine its responses based on user interactions.

54. **Q: How does ReAct prompting benefit user interactions?**

    A: It enhances the model's ability to generate relevant and contextually appropriate responses, improving user satisfaction and task performance.

10. Miscellaneous Topics in Generative AI

55. **Q: What is transfer learning, and how is it applied in Generative AI?**

    A: Transfer learning uses a pre-trained model on a new, related task. In Generative AI, models like GPT-3 are fine-tuned for specific tasks with smaller datasets.

56. **Q: How does GPT differ from BERT?**

    A: GPT (Generative Pre-trained Transformer) is designed for text generation with a unidirectional approach, while BERT uses a bidirectional approach for understanding context, excelling in tasks like question answering and text classification.

57. **Q: What is BERT, and how is it used in Generative AI?**

    A: BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained model designed for understanding context by considering both left and right surroundings in text. While not generative itself, BERT's architecture is adapted for text generation tasks in models like GPT.

58. **Q: What is style transfer in Generative AI?**

    A: Style transfer applies the style of one image to the content of another by separating and recombining their content and style representations using neural networks.

59. **Q: What is reinforcement learning, and is it used in Generative AI?**

    A: Reinforcement learning trains an agent to make decisions by rewarding desired actions and penalizing undesired ones. Though not typically used in generative tasks, it can be combined with generative models in game development and optimizing content generation strategies.

60. **Q: What is a Deepfake?**

    A: A Deepfake is synthetic media created using Generative AI techniques, such as GANs, to produce realistic but fake representations of people.

11. Commonly used AI Algorithms

61. **Q: What is a Generative Adversarial Network (GAN) and how does it work?**

    A: A Generative Adversarial Network (GAN) consists of two neural networks: a generator and a discriminator. The generator creates fake data samples, while the discriminator evaluates whether the samples are real (from the training set) or fake (created by the generator). They are trained simultaneously in a zero-sum game, where the generator improves at creating realistic data, and the discriminator gets better at detecting fakes. This adversarial process continues until the generator produces highly realistic data that the discriminator can no longer distinguish from real data.

62. **Q: How does a Variational Autoencoder (VAE) differ from a traditional autoencoder?**

    A: A Variational Autoencoder (VAE) differs from a traditional autoencoder in that it learns a probabilistic latent space instead of a deterministic one. In a VAE, the encoder maps input data to a distribution (typically Gaussian) in the latent space, rather than a single point. The decoder then samples from this distribution to reconstruct the data. This allows the VAE to generate new data by sampling from the learned latent space, making it useful for generating new, similar data points.

63. **Q: Can you explain the architecture of a Transformer model?**

    A: The Transformer model architecture, introduced in the paper "Attention is All You Need," consists of an encoder-decoder structure. The encoder is composed of multiple identical layers, each with two sub-layers: a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. The decoder also has similar layers but includes an additional sub-layer for multi-head attention over the encoder's output.

The model uses positional encodings to maintain the order of input sequences and relies heavily on self-attention mechanisms to process data in parallel, making it highly efficient for tasks like language translation and text generation.

64. **Q: What is the self-attention mechanism in Transformers, and why is it important?**

    A: The self-attention mechanism in Transformers allows the model to weigh the importance of different words in a sentence relative to each other. It computes attention scores by comparing each word in the sequence with every other word, helping the model to capture contextual relationships and dependencies. This mechanism is crucial because it enables the model to understand the context more effectively than traditional RNNs or CNNs, leading to improved performance in tasks like machine translation, text summarization, and language modeling.

65. **Q: How do Recurrent Neural Networks (RNNs) differ from Transformers in handling sequential data?**

    A: Recurrent Neural Networks (RNNs) process sequential data by maintaining a hidden state that captures information from previous time steps, which is updated as new data is processed sequentially. This makes RNNs inherently sequential and less parallelizable. Transformers, on the other hand, use self-attention mechanisms that allow them to process entire sequences in parallel, making them more efficient and capable of capturing long-range dependencies without the vanishing gradient problem that RNNs often face.

66. **Q: What is the purpose of the Encoder-Decoder architecture in NLP models?**

    A: The Encoder-Decoder architecture is designed to handle tasks where the input and output sequences can be of different lengths, such as machine translation. The encoder processes the input sequence and encodes it into a fixed-size context vector, capturing the relevant information. The decoder then takes this context vector and generates the output sequence. This architecture allows the model to generate outputs that are contextually relevant and coherent with respect to the input data.

67. **Q: How does the Attention mechanism enhance the performance of Transformer models?**

    A: The Attention mechanism enhances the performance of Transformer models by allowing them to focus on different parts of the input sequence when generating each word in the output sequence. It calculates a weighted sum of input values, where the weights are determined by the relevance of each input value to the current output being generated. This allows the model to capture dependencies regardless of their distance in the sequence, leading to better performance in understanding and generating coherent and contextually accurate text.

68. **Q: What are Bidirectional Encoder Representations from Transformers (BERT), and how are they used in NLP?**

    A: BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model designed for understanding context in NLP tasks. It uses a bidirectional approach, meaning it considers both left and right context simultaneously during training. This enables BERT to capture richer contextual information compared to unidirectional models. BERT is used in various NLP tasks such as question answering, sentiment analysis, and text classification by fine-tuning it on specific datasets relevant to the task.

69. **Q: How does GPT (Generative Pre-trained Transformer) differ from BERT?**

    A: GPT (Generative Pre-trained Transformer) and BERT differ primarily in their training objectives and architecture. GPT is designed for text generation and is trained using a unidirectional approach, meaning it predicts the next word in a sequence based on the previous words. BERT, on the other hand, uses a bidirectional approach and is pre-trained with a masked language modeling objective, where it predicts missing words in a sentence based on both left and right context. GPT excels in generative tasks, while BERT is optimized for understanding context in tasks like classification and question answering.

70. **Q: What is style transfer in Generative AI, and how does it work?**

    A: Style transfer in Generative AI is a technique that involves applying the style of one image (e.g., a painting's brush strokes) to the content of another image (e.g., a photograph). This is achieved by separating and recombining the content and style representations of the images using neural networks. Typically, a convolutional neural network (CNN) is used to extract content features from the target image and style features from the style image. Then, an optimization process blends these features to produce an image that maintains the content of the target image while adopting the style characteristics of the style image.

71. **Q: How do sequence-to-sequence models work, and what are their applications?**

    A: Sequence-to-sequence (Seq2Seq) models consist of an encoder-decoder architecture used to map input sequences to output sequences, often of different lengths. The encoder processes the input sequence into a fixed-size context vector, which is then passed to the decoder to generate the output sequence. Applications of Seq2Seq models include machine translation, text summarization, and conversational agents.

72. **Q: What is the difference between a convolutional neural network (CNN) and a recurrent neural network (RNN)?**

A: CNNs and RNNs are designed for different types of data. CNNs are primarily used for grid-like data such as images, where they apply convolutional layers to capture spatial hierarchies. RNNs, on the other hand, are designed for sequential data, such as time series or text, where they maintain a hidden state to capture temporal dependencies. While CNNs excel in image-related tasks, RNNs are typically used for tasks involving sequences.

73. **Q: How does transfer learning benefit Generative AI models?**

    A: Transfer learning benefits Generative AI models by leveraging pre-trained models on large datasets and fine-tuning them for specific tasks with smaller datasets. This approach significantly reduces training time and resource requirements while improving performance, as the pre-trained models already have learned useful features and representations from the large initial datasets.

74. **Q: What are attention heads in the context of Transformer models?**

    A: Attention heads are components of the multi-head self-attention mechanism in Transformer models. Each attention head performs an independent self-attention operation, capturing different aspects of the relationships between words in a sequence. The outputs of multiple attention heads are then concatenated and linearly transformed to produce the final attention output. This allows the model to capture a richer set of dependencies and contextual information.

75. **Q: How do autoencoders contribute to data compression and generation in Generative AI?**

    A: Autoencoders contribute to data compression by learning to encode input data into a lower-dimensional latent space and then decoding it back to reconstruct the original data. This encoding-decoding process reduces dimensionality and captures the most important features of the data. In Generative AI, the learned latent space can be used to generate new data samples by sampling from it and decoding the samples into the original data space, enabling tasks like data augmentation and synthetic data generation.

76. **Q: What is the role of beam search in text generation models?**

    A: Beam search is a decoding algorithm used in text generation models to find the most likely sequence of words. It maintains multiple candidate sequences (beams) at each step and expands them by considering the most probable next words. Beam search balances between exploring diverse possibilities and focusing on high-probability sequences, improving the quality and coherence of the generated text compared to greedy search.

77. **Q: How does the GPT model handle long-range dependencies in text?**

    A: The GPT model handles long-range dependencies in text using the self-attention mechanism, which allows it to weigh the importance of each word in the input sequence relative to every other word. This enables the model to capture relationships and dependencies over long distances within the text, improving its ability to generate coherent and contextually relevant text.