Q1 Answer:

The domain generalization ability of LLMs, is indeed influenced by multiple factors including the below:

Model Architecture: Transformer architecture-based LLMs have been very successful. Their self-attention mechanism allows them to capture long-range dependencies and contextual information. Also, deeper and wider models can learn more complex patterns and representations. These aspects are crucial for domain generalization.

Training Data: The diversity and quality of the training data matters. Models trained on a wide variety of sources generalize better because of their diverse exposure. A data preprocessing step filtering out poor quality text, removing biases, ensuring balance across domains etc. helps to avoid overfitting specific features of seen domains and to generalize better.

Fine-Tuning Techniques: Fine-tuning on domain-specific datasets can improve performance in those domains but can sometimes reduce the ability to generalize. However, Fine-tuning on multiple tasks simultaneously (multi-task fine-tuning) can help the model learn shared representations across domains. Additionally, techniques like RLHF & PPO, DPO can help make the generated text more aligned to human preference.

Q2 Answer:

In GANs, the generator aims to create realistic data samples, which the discriminator cannot distinguish in comparison with real samples. Ideally, the generator should produce a wide variety of outputs resembling the diversity of the training data. However, during mode collapse, the generator produces only a limited/repetitive outputs, essentially "collapsing" to a few modes/patterns.

The most noticeable symptom is the lack of diversity in the generated outputs even for different inputs. For example, a GAN which was trained to generate images of animals, predominantly generates images of limited types of animals only while not generating much of the rest.

While the quality of individual outputs might still be high, the overall usefulness of the GAN is compromised because it cannot generate a diverse set of samples capturing full variability present in the real data.

Q3 Answer:

While both Variational Autoencoders (VAEs) and standard autoencoders aim to learn efficient data representations, they differ in their approach.

Standard autoencoders use a deterministic approach in their encoding-decoding process. Meaning, the encoder maps inputs to fixed points in the latent space which allows decoder to reconstruct the original inputs only but not generate new samples.

On the contrary, VAEs introduce a probabilistic element. The encoder outputs parameters of a probability distribution rather than a fixed point in the latent space which allows for sampling new data instances.

Q4 Answer:

The runtime complexity of transformer-based text encoder models is significantly influenced by their self-attention mechanism, which allows each token in the input sequence to attend to every other token, capturing dependencies regardless of their position in the sequence. This results in quadratic computational complexity in terms of sequence length.

As the sequence length increases, the computational cost as well as memory requirements grow quadratically, making them computationally expensive and time-consuming. This limits the scalability of transformer models for tasks involving long sequences such as document summarization.

To overcome this, several techniques like sparse attention, hierarchical attention, modified self-attention (to achieve linear/sub-quadratic complexity) are often applied.