PassQueen
a.k.a dumpsqueen.com

NVIDIA Generative AI LLMs

NVIDIA NCA-GENL

Version Demo

Total Demo Questions: 13

Total Premium Questions: 135

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Generative AI and LLM Fundamentals	50
Topic 2, Prompt Engineering	9
Topic 3, Data Preparation and Model Fine-Tuning	35
Topic 4, LLM Integration and Deployment	39
Topic 5, Mix Questions	2
Total	135

QUESTION NO: 1

Your company has upgraded from a legacy LLM model to a new model that allows for larger sequences and higher token limits. What is the most likely result of upgrading to the new model?

A. The number of tokens is fixed for all existing language models, so there is no benefit to upgrading to higher token limits.

B. The newer model allows for larger context, so the outputs will improve without increasing inference time overhead.

C. The newer model allows the same context lengths, but the larger token limit will result in more comprehensive and longer outputs with more detail.

D. The newer model allows larger context, so outputs will improve, but you will likely incur longer inference times.

ANSWER: D

Explanation:

The newer model allows larger context, so outputs will improve, but you will likely incur longer inference times. A larger sequence length, often called a larger context window, lets the model consider more of a prompt, document, conversation history, or retrieved material in a single request. This can improve the relevance, coherence, and completeness of results when the task depends on information distributed across a longer input—for example, summarizing lengthy documents, answering questions over multiple retrieved passages, or maintaining multi-turn conversational context.

That capability normally has a performance cost. Transformer inference must retain key-value cache data for prior tokens during autoregressive generation, so cache memory grows as the context grows. Processing more tokens also requires additional computation, which can increase latency and/or lower throughput when using the same hardware and serving configuration. The exact impact depends on model architecture, prompt and output lengths, batching, hardware, and inference optimizations, but longer inference times are the expected practical trade-off unless capacity or optimizations are added. NVIDIA discusses transformer performance and memory-efficient execution techniques in the NVIDIA Transformer Engine User Guide; the relationship between generated-token history and memory use is also described in the Transformers cache explanation.

QUESTION NO: 2

In the context of preparing a multilingual dataset for fine-tuning an LLM, which preprocessing technique is most effective for handling text from diverse scripts (e.g., Latin, Cyrillic, Devanagari) to ensure consistent model performance?

A. Normalizing all text to a single script using transliteration.

B. Applying Unicode normalization to standardize character encodings.

C. Removing all non-Latin characters to simplify the input.

D. Converting text to phonetic representations for cross-lingual alignment.

ANSWER: B

Explanation:

Applying Unicode normalization to standardize character encodings is the appropriate preprocessing step for multilingual LLM fine-tuning. Unicode permits some visually identical text to be represented by distinct sequences of code points—for example, a character may appear as one precomposed code point or as a base character followed by a combining mark. Data collected from different sources, operating systems, keyboards, and languages can therefore contain inconsistent representations of equivalent text. Without normalization, these variations can create avoidable vocabulary and tokenization differences, fragment equivalent examples, and increase data sparsity.

Normalizing text consistently, commonly with NFC when preserving canonical distinctions is important, makes equivalent character sequences conform to a standard representation while retaining the original writing systems. This supports stable tokenizer behavior and cleaner training data across Latin, Cyrillic, Devanagari, and other scripts. The selected normalization form should be applied consistently to both training and inference inputs, with validation that it does not conflict with language-specific requirements. Unicode's normalization standard defines the canonical and compatibility normalization

forms and their intended use. NVIDIA NeMo's NLP tooling is designed for multilingual and language-model workflows where robust, consistent text preparation is foundational.

References: Unicode Standard Annex #15: Unicode Normalization Forms; NVIDIA NeMo NLP Overview.

QUESTION NO: 3

Which of the following practices support reliable evaluation of an LLM application before production deployment? (Pick the 2 correct responses)

A. Using a representative evaluation dataset.

B. Defining measurable quality and safety criteria.

C. Evaluating only on the data used for fine-tuning.

D. Changing success criteria after each model response.

ANSWER: A B

Explanation:

Using a representative evaluation dataset is correct because the dataset should reflect the intended users, tasks, domains, and realistic input patterns of the application. Evaluation results on unrepresentative examples may not predict performance after deployment.

Defining measurable quality and safety criteria is also correct. Teams should establish criteria such as task correctness, groundedness, format compliance, refusal behavior, latency, and harmful-output rates before comparing system versions. Clear criteria make regressions visible and support repeatable release decisions.

Reliable evaluation combines appropriate test data, defined metrics, and systematic comparison of model or application changes. NVIDIA NeMo provides evaluation capabilities for language-model workflows, and NVIDIA NeMo Guardrails supports testing and evaluation of guardrail behavior. See the NVIDIA NeMo LLM evaluation documentation and NVIDIA NeMo Guardrails evaluation documentation.

QUESTION NO: 4

What metrics would you use to evaluate the performance of a RAG workflow in terms of the accuracy of responses generated in relation to the input query? (Choose two.)

A. Generator latency

B. Retriever latency

C. Tokens generated per second

D. Response relevancy

E. Context precision

ANSWER: D E

Explanation:

Response relevancy is an appropriate metric because it evaluates whether the generated answer directly addresses the user's input query. In a retrieval-augmented generation workflow, an accurate response must be relevant to the intent and information requested by the user, rather than merely being fluent or well formed. This metric is commonly assessed using human evaluation or an LLM-based evaluator that compares the response's content with the original question.

Context precision is also appropriate because response accuracy depends on retrieving evidence that is genuinely relevant to the query. Context precision measures the proportion of retrieved context items that are useful and relevant among the

returned results. High context precision gives the language model better grounding material, which supports generation of answers that are pertinent and factually supported by the retrieved knowledge. Together, these metrics assess both the answer's relationship to the query and the quality of the retrieved supporting context, which are central dimensions of RAG quality evaluation. NVIDIA's RAG evaluation guidance and the Ragas metric documentation describe these types of generation and retrieval quality measurements: NVIDIA NeMo Guardrails evaluation documentation and Ragas metrics overview.

QUESTION NO: 5

Which of the following claims is correct about quantization in the context of Deep Learning? (Pick the

2 correct responses)

A. Quantization might help in saving power and reducing heat production.

B. It consists of removing a quantity of weights whose values are zero.

C. It leads to a substantial loss of model accuracy.

D. Helps reduce memory requirements and achieve better cache utilization.

E. It only involves reducing the number of bits of the parameters.

ANSWER: A D

Explanation:

Quantization might help in saving power and reducing heat production because it converts model values and computations from higher-precision representations, such as FP32, to lower-precision formats such as FP16, INT8, or INT4. Lower-precision inference reduces memory movement and can use hardware units optimized for these data types. Since memory access and arithmetic operations consume energy, this can lower power consumption and associated thermal output, particularly for high-throughput inference or deployment on resource-constrained systems.

Helps reduce memory requirements and achieve better cache utilization is also correct. Storing weights and, where applicable, activations in fewer bits reduces the memory footprint: INT8 values require one quarter of the storage of FP32 values. A smaller model requires less memory bandwidth and enables more parameters to reside in caches, which reduces data-transfer overhead. These effects can improve inference efficiency, throughput, and latency when the hardware and inference runtime support the selected precision. NVIDIA TensorRT documents quantization as an approach for executing inference using reduced precision while retaining suitable model quality, and its optimization workflow includes calibration or quantization-aware methods to establish appropriate numeric ranges. See the NVIDIA TensorRT quantization documentation and the TensorRT Developer Guide.

QUESTION NO: 6

In the Transformer architecture, which of the following statements about the Q (query), K (key), and V (value) matrices is correct?

A. Q, K, and V are randomly initialized weight matrices used for positional encoding.

B. K is responsible for computing the attention scores between the query and key vectors.

C. Q represents the query vector used to retrieve relevant information from the input sequence.

D. V is used to calculate the positional embeddings for each token in the input sequence.

ANSWER: C

Explanation:

Q represents the query vector used to retrieve relevant information from the input sequence. In a Transformer attention layer, the model forms query, key, and value representations by applying separate learned linear projections to token

representations. A query encodes what the current token or position is seeking; keys encode the matching characteristics available at all positions; and values hold the information that will be combined into the output representation.

For scaled dot-product attention, each query is compared with keys using a dot product. The resulting scores are scaled by the square root of the key dimension and normalized with softmax: Attention(Q, K, V) = softmax($QK^T/\sqrt{d_k}$)V. This produces attention weights that identify the sequence positions most relevant to the query. Those weights are then applied to the value vectors, yielding a weighted combination of information from the selected positions. Thus, the query is accurately described as the representation used to retrieve relevant sequence information through its similarity relationship with keys.

This query-key-value formulation is defined in the original Transformer architecture and is foundational to modern LLM attention mechanisms. See Attention Is All You Need and NVIDIA's NeMo NLP Overview.

QUESTION NO: 7

What is the primary objective of supervised fine-tuning (SFT) for a large language model?

A. To adapt a pretrained model using examples of desired input-output behavior.

B. To convert model weights from FP16 to INT8.

C. To retrieve documents from an external knowledge base.

D. To increase the context window without additional training.

ANSWER: A

Explanation:

To adapt a pretrained model using examples of desired input-output behavior is correct. Supervised fine-tuning continues training from a pretrained foundation model on curated examples containing prompts and target responses. The training objective encourages the model to generate outputs that match the demonstrated answers, formats, style, and task requirements.

SFT is commonly used to specialize a general-purpose model for a domain or task such as customer support, information extraction, summarization, or instruction following. Unlike pretraining, which learns broad language patterns from large-scale text corpora, SFT uses labeled demonstrations to align model behavior with a specific desired response. NVIDIA NeMo supports supervised fine-tuning workflows for adapting language models to downstream use cases. See the NVIDIA NeMo supervised fine-tuning guide.

QUESTION NO: 8

Which model deployment framework is used to deploy an NLP project, especially for highperformance inference in production environments?

A. NVIDIA DeepStream

B. HuggingFace

C. NeMo

D. NVIDIA Triton

ANSWER: D

Explanation:

NVIDIA Triton is the appropriate framework for deploying an NLP project when the objective is high-performance production inference. NVIDIA Triton Inference Server is designed to serve AI models efficiently at scale, including NLP and large-language-model workloads. It supports models from common ecosystems such as PyTorch, TensorFlow, ONNX Runtime,

TensorRT, and Python-based backends, allowing teams to package trained NLP models behind standardized HTTP or gRPC inference endpoints.

Its production-focused capabilities are particularly relevant to demanding NLP services: dynamic batching combines compatible requests to improve GPU utilization and throughput, concurrent model execution helps use available compute effectively, and model repositories provide controlled loading and version management. Triton also integrates with NVIDIA TensorRT and NVIDIA GPUs to reduce latency and maximize inference performance. These characteristics make it a core NVIDIA serving component when moving a trained language model from development into a scalable production environment.

For implementation details and supported model backends, consult the NVIDIA Triton Inference Server User Guide. NVIDIA also summarizes its production-serving capabilities on the Triton Inference Server product page.

QUESTION NO: 9

Which of the following practices help defend an LLM application against prompt-injection attacks? (Pick the 2 correct responses)

A. Separating untrusted content from application instructions.

B. Applying authorization checks before executing tool actions.

C. Allowing retrieved documents to override system instructions.

D. Automatically executing every tool call generated by the model.

ANSWER: A B

Explanation:

Separating untrusted content from application instructions is correct because retrieved documents, web pages, and user-provided text must not be treated as authoritative instructions. Clearly delimiting untrusted data and keeping application policies in trusted instruction channels reduces the risk that embedded malicious text changes the intended task.

Applying authorization checks before executing tool actions is also correct. An LLM output should not by itself grant access to data, send messages, modify records, or invoke other consequential operations. The application should validate requested actions and enforce identity, permissions, and policy controls independently of generated text.

Prompt injection is an application-security risk that requires defense in depth, including trusted instruction handling, input controls, constrained tools, and output validation. See the OWASP guidance on prompt injection and NVIDIA's NeMo Guardrails documentation.

QUESTION NO: 10

What is the prompt œTranslate English to French: cheese => an example of?

A. Few-shot learning

B. Fine tuning a model

C. One-shot learning

D. Zero-shot learning

ANSWER: D

Explanation:

“Zero-shot learning” is correct because the prompt gives the model a task instruction—translate from English to French—and supplies the item to translate, “cheese,” without including any input-output translation demonstrations in the prompt. The model is expected to infer the requested behavior from the instruction and apply knowledge acquired during pretraining,

producing an answer such as “fromage.” In prompt-engineering terminology, this is zero-shot prompting: a model receives an instruction and a new query but no examples that illustrate the desired task format or output behavior in the current context. This capability is important for LLM applications because well-written natural-language instructions can enable useful task performance without task-specific retraining or in-context examples. NVIDIA describes prompt engineering as providing instructions and context that guide an LLM’s output; zero-shot prompting is the minimal form of this approach, relying on the instruction itself. For further background, see NVIDIA’s LLM prompting documentation and the Prompt Engineering Guide’s zero-shot prompting overview.

QUESTION NO: 11

What is the prompt “Translate English to French: cheese = > ” an example of?

A. Few-shot learning

B. Fine tuning a model

C. One-shot learning

D. Zero-shot learning

ANSWER: D

Explanation:

Zero-shot learning is correct because the prompt instructs a pretrained language model to perform a translation without supplying any input-output examples that demonstrate the requested task. The model must infer the task from the natural-language instruction, “Translate English to French,” and use knowledge acquired during pretraining to generate the French word for “cheese.” In prompt engineering terminology, zero-shot prompting provides only a task description or instruction; it does not include demonstrations in the prompt context. This capability is one of the practical strengths of large language models: a single, clearly expressed instruction can often invoke a task that was not separately trained or configured for the user. Translation is a common example because the model can apply its learned multilingual representations directly from the stated instruction. NVIDIA’s Generative AI learning materials describe zero-shot prompting as directing an LLM to complete a task without providing examples, while examples can be added when a task needs more guidance. For further background on NVIDIA’s LLM learning content, see NVIDIA Generative AI and LLMs course and the NVIDIA Learning and Models documentation.

QUESTION NO: 12

Which of the following optimizations are provided by TensorRT? (Choose two.)

A. Data augmentation

B. Variable learning rate

C. Multi-Stream Execution

D. Layer Fusion

E. Residual connections

ANSWER: C D

Explanation:

TensorRT provides **Layer Fusion** and **Multi-Stream Execution** to improve inference efficiency on NVIDIA GPUs. During engine building, TensorRT analyzes the network graph and fuses compatible operations into optimized kernels where possible. For example, a sequence involving a convolution, bias addition, and activation can be executed with less intermediate memory traffic and fewer kernel launches than separate operations. This graph-level optimization reduces overhead and makes better use of GPU compute and memory bandwidth.

Multi-Stream Execution supports efficient concurrent inference activity through CUDA streams. In serving scenarios, independent work can be issued asynchronously, allowing the GPU to overlap eligible operations and maintain higher utilization. TensorRT also supports mechanisms such as auxiliary streams for parallelizable portions of an engine's execution, subject to dependency and hardware constraints. These capabilities are particularly useful for throughput-oriented deployments handling multiple requests or batches.

Both capabilities concern optimizing and scheduling an already-trained network for deployment rather than changing its training procedure or model architecture. NVIDIA describes TensorRT as an SDK for high-performance deep-learning inference that applies graph optimizations and uses CUDA execution facilities to accelerate deployed models. See the TensorRT Architecture Overview and NVIDIA's guidance on TensorRT Performance Best Practices.

QUESTION NO: 13

Which of the following are important operational metrics for monitoring an LLM inference service? (Pick the 2 correct responses)

A. End-to-end request latency.

B. Number of source-code comments in the application.

C. Tokens generated per second.

D. Number of colors in the user interface.

E. Length of the model card title.

ANSWER: A C

Explanation:

End-to-end request latency is correct because it measures the time a client waits from sending a request until receiving the completed response. Monitoring latency helps teams determine whether the service meets user-facing service-level objectives and identify degradation caused by traffic, long prompts, resource contention, or configuration changes.

Tokens generated per second is also correct because it measures generation throughput during decoding. This metric is particularly relevant for streaming LLM applications, where users perceive output responsiveness as tokens are produced. It can help identify performance changes after altering hardware, batching, model precision, or inference-engine settings. See the NVIDIA Triton Inference Server metrics documentation and the NVIDIA TensorRT-LLM performance documentation.