Scale and Deploy LLMs in Production Environments

Course Overview

Hi, everyone. My name is Abhishek Kumar, and welcome to my course, Scale and Deploy LLMs in Production Environments. I’m a data science consultant, speaker, author, and graduate of UC Berkeley. LLMs have truly revolutionized the world and have opened newer and exciting opportunities, but scaling and deploying these LLMs can be challenging and hard, and in this course, you will learn about various considerations and blueprints to take the full advantage of these LLMs in enterprise production settings. Some of the major topics that we will cover include approaches for LLM deployments, LLM observability and monitoring, LLM as APIs and integration with enterprise ecosystem, UX considerations, and making LLMs secure and compliant. By the end of this course, you will know everything you need to create a tailored blueprint and strategy for your own organizational LLM applications at scale to create a competitive edge. Before beginning the course, you should be familiar with LLM fundamentals. I hope you will join me on this journey to learn to design and launch exciting production grade LLM‑powered applications with the Scale and Deploy LLMs in Production Environments course, at Pluralsight.

Deploying LLMs: Blueprints and Considerations

Approaches for LLMs in Production

Large language models are revolutionizing industries and creating vast opportunities. However, they are of little value unless deployed and accessible to end users and downstream services, and deploying and scaling these LLMs require careful considerations aligned with organizational and use case needs. The approaches for LLMs in production can vary according to different types of use cases. Carved Rock Fitness, an online retailer for adventure sports gear, wants to use generative AI and large language models for three different use cases, so let’s explore how the LLM approaches and blueprints would vary.

As the first use case, the company wants to use LLM *to send personalized emails to their existing customers for upselling and cross‑selling opportunities*. For this use case, the company wants to leverage different data signals, such as user demographics, purchase history, and shopping cart information to create personalized emails. For this implementation, the team can use a prompt engineering blueprint to feed input signals to a large language model and generate personalized email content. Using a third‑party LLM API provider like OpenAI, Anthropic can help you easily set up this blueprint, and it can be cheaper also if there is less usage. You can work with iterative prompt template to generate content that suits your purpose. However, this approach may not be suitable for complex tasks and quality of generated text may not be reliable. Due to limitations of context window of LLMs, you can only feed limited inputs as prompt. Additionally, if there is a plan to leverage self‑hosted LLMs, then it will further add to the complexity.

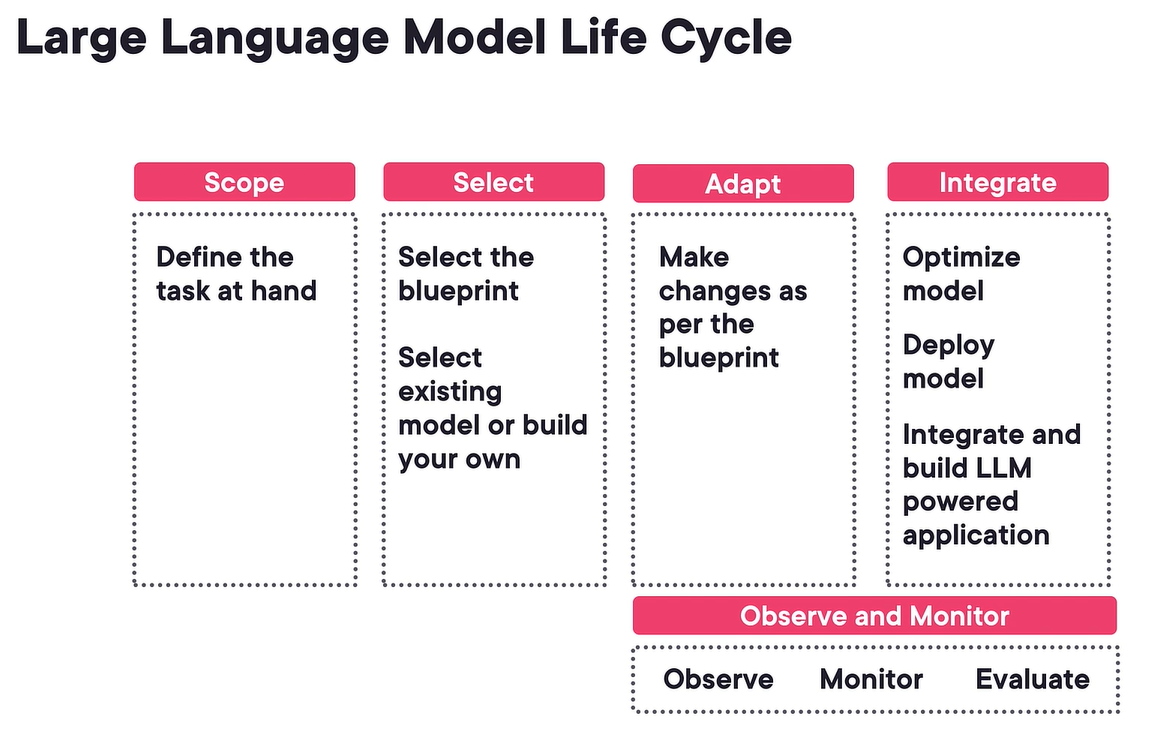
As a second use case, the Carved Rock Fitness Company aims *to implement chat bots for customer service on their ecommerce platform* to assist customers with product‑related queries. Using only prompt engineering may not be sufficient for this use case. Since the LLM has not been trained on the private or recent data about the products and manuals of the fitness company leading to suboptimal responses and bad customer experience. To tackle this issue, we can use another blueprint using **retrieval augmented generation**, or RAG. There, we can encode enterprise data using embedding models and save it to vector databases called index stores, and then when you have a query, using the retriever module to select relevant information from the index store and adding this information to the LLM context along with the query and prompt template to generate an appropriate response. The RAG approach allows for inclusion of private and recent enterprise data to LLM calls optimizing the token usage and reducing hallucinations. This approach shows promising results and increased confidence in LLM‑generated outcomes by sourcing citations and providing attribution.

However, this approach has *drawbacks* too as you will need to build different components of the RAG pipeline on your own, which can be challenging. Maintaining response tones and understanding domain nuances can also be difficult with base LLMs, and self‑hosting and managing LLMs will add to the complexity if you are opting for it.

The third use case for Carved Rock Fitness is *to analyze voice of customers on the feedback and the reviews data that they capture on their platforms and discussion forums*, and they found out that available LLMs are not sufficient for their purpose, these models are not customized for the tasks or they do not understand the domain‑specific nuances, and the generated response tones are inconsistent. So another blueprint is recommended for this use case that requires **fine tuning**. To achieve a fine‑tuned LLM, the available dataset must be processed and shaped for fine tuning using a base LLM. Once the fine tuning is done, then the input along with the relevant prompt template can be fed to the fine-tuned LLM to generate appropriate response. This approach allows to adapt the LLM for a specific task and domain, and using base LLM that is already trained on vast amount of data allows tuning with limited data. Also a fine‑tuned model with learned patterns requires fewer tokens during inference, thereby reducing inference cost.

This approach has its own *challenges*, including dataset preparation, cost of fine tuning, and potential overfitting. Catastrophic forgetting can also occur with suboptimal fine tuning that can affect LLM ability to perform different tasks. Also, a strong evaluation framework is needed for robust fine‑tuned models.

So we did look at three different blueprints to enable Carved Rock Fitness, but similar blueprints can be leveraged by any other organization, and if you put everything in a LLM lifecycle, the typical approach would be to first identify the scope based on the task at hand. Accordingly, you can select one of the blueprint. Based on the blueprint, you adapt the LLM for your task, and once you have a model in the appropriate shape, you further optimize it to reduce the compute and memory footprint, then you deploy it and integrate it with your application and rest of the organizational ecosystem. You need to also ensure that you are observing, monitoring, and evaluating these LLMs at every step of the journey. In fact, due to specific nuances and the specialization of managing the LLM lifecycle, a new field has emerged called LLMOps, and we will look at various components of LLMOps and associated considerations further in this course.



COTS LLM APIs vs. Self-hosted LLMs

In the large language model lifecycle, we saw that you have to first define the scope based on the task you want to accomplish using the LLM. Assuming you know the use case and the task, the next very critical step is to pick the blueprint and select whether you want to use existing model or build your own. The key question you will encounter as to how to choose the LLM for the task, and whether to use a third‑party LLM provider or manage the LLM on your own. Well, there are many commercial off‑the‑shelf, or COTS, LLM APIs available nowadays. Just to name a few popular ones at the time of recording, are OpenAI, Azure OpenAI, Google Vertex AI, AWS Bedrock, and Anthropic, and these LLM APIs might give you a head start as you don’t have to worry about the infra and the deployment as it is already taken care by the API providers. This approach might even be cheaper if you’re not expecting huge number of API calls, and these APIs might be sufficient for most of the common use cases as well.

However, there could be limitations with these APIs.

The first key one is data security as you’re relying on a third‑party API provider. While many of these APIs do provide data security commitments and 0 data retention options, for some industries, sending data to a third‑party API provider can be a deal breaker.

There can also be geo or region‑specific regulations. These COTS APIs might also be limited for specific customization that your task, domain, and scenario requires.

Performance restrictions can also be another key factor. The API provider have their own SLAs and rate limits, but if your requirement demands different SLAs for non‑functional requirements, or NFRs, such as latency and throughput, you will have to find alternatives.

Cost could be another consideration. While on a very high level, these APIs might look cheap, but based on your usage, the cost can blow up very quickly. For example, for the Carved Rock Fitness company personalized email use case, let’s say that for a generation of one mail, 4,000 words or tokens are used, taking an average API pricing of $0.03 per 1,000 token, generation of one mail with this API would cost $0.12. And if this company send mails to its 1 million users per week, the cost would be $0.12 million per week and 6.24 million per year for 52 weeks, and if this company scales to 10 million users, you are looking at a whopping $62.4 million LLM API bill per year.

In view of all of the considerations, self‑hosted and managed LLMs can be a great alternative. There are multiple base **open source LLMs** available to choose from. You can check out the popular HuggingFace platform to find many of these open source LLMs, such as Bloom, Falcon, Lama, Mistral, and Yi. There are many many more, and I would highly encourage you to look at the HuggingFace LLM leaderboard to find the latest top models, and by hosting and managing these LLMs on your own, you can manage your data security and regulation needs. You can further customize these models as per your requirement. You can scale up or down the infrastructure as per your organizational SLA, and you can lower the cost to get better return on investment. Further, this approach also allows you to avoid any vendor lock‑in as you can switch between different models if you want to, and given the open-source ecosystem of LLMs is growing very rapidly, you can benefit from this evolving space. With each passing week, more and more efficient models are becoming available.

Well, this approach has its own set of challenges that you should be aware of before you venture into this option. This approach requires considerable technical skills to deploy such models, and if you have not set up properly, you might get subpar performance, and you will have to manage the lifecycle of these models that can be an overhead, and cost benefit will take a hit if these systems are not well planned and executed.

Self-hosted LLMs: Cloud vs. Edge

* Switching back to our LLM lifecycle diagram, let’s say that you have selected the path of self‑hosted model, and you have adjusted the blueprint based on your requirement such as finetuning the model.
* Then the next set of steps would be to optimize this model and deploy it, so you will have to make another decision, that is whether you need to deploy the LLM in the cloud infrastructure or on the edge devices or nodes.
  + For deploying on Cloud, you take your LLM and package and ship it to cloud infrastructure that will be eventually consumed by the end user or system devices, and this is going to be the most common pattern for the organizations, and we are going to discuss more about this approach in upcoming sections,
  + But let’s look at the alternative option where you take a large language model and deploy the quantized or compressed version directly on the Edge devices, and there could be several reasons why you’d want your LLMs on Edge, maybe you have a scenario with limited network connectivity or your use case requires ultra-low latency and API calls to the server might not work in such cases. LLM on Edge can also be suited for scenarios where data cannot leave the device due to privacy and security concerns. LLM on Edge can also be used to provide a hyperpersonalized experience by working directly on data signals available on the devices, and since the devices are becoming computationally powerful these days, such an approach can provide a highly scalable option.

There are some interesting use cases for Edge deployment of these LLMs. Consider remote factories and rigs with limited network bandwidth. By bringing the LLM models to the Edge, newer opportunities can open up. Similarly, due to ultra-low latency on device patterns can be learned quickly to provide a truly LLM augmented personalized experience, and here you can also provide an active feedback loop to generate better responses from these LLMs much faster, however, Edge deployment comes with its own set of challenges as well. Based on the LLM model size, you may encounter compute or memory resource constraints, and running these models on Edge devices also mean that they will require more battery or other energy sources, and if not deployed properly, these models available on the devices can be gamed or hacked. Device compatibility is another significant challenge to solve as not all the Edge devices are same and your Edge deployment strategy will have to take this into account.

LLM Observability and Monitoring

LLM observability and monitoring is another very important aspect in the LLM lifecycle. We all know that LLMs are prone to hallucinate, and the responses can vary significantly by changing different LLMs. So during the model selection and tuning phase, it is very important to monitor and evaluate effectiveness of these models, and there are a bunch of evaluation metrics that you can look at. For example, *answer relevancy* to understand how relevant the generated response is relative to the input, *contextual precision* is commonly used metric in the retrieval augmented generation or rag system to assist the quality of the retriever module in order to check how relevant is the context pulled from the index store. *Faithfulness* is another metric to factually align the response to the available context. Some evaluation frameworks like Galileo have their own take on providing a holistic hallucination index.

You can also leverage **different evaluation benchmarks**, such as *HellaSwag* that evaluates the LLM ability to complete a sentence, *TruthfulQA* that measures the truthfulness of model response, and *MMLU* that assesses the LLM ability to multitask.

There are **different evaluation frameworks** that you can explore. For example, *OpenAI eval framework* is the popular one. *Galileo* offers a commercial LLM evaluation solution. *Ragas* is another evaluation framework, especially for RAG applications, and these evaluation and monitoring is not just limited to model selection and the finetuning phase. Once you have integrated the LLM in your application, you will have to continuously monitor, observe, and evaluate, not only the *functional requirements such as response quality and hallucinations*, but also *non‑functional requirements, such as cost, latency, throughput, and availability*, and you can use end‑to‑end **LLM observability and monitoring tools and platforms** to accomplish these goals. At the time of recording, some of the most popular ones are *LangSmith by LangChain, Arize ML, WandB, and Whylabs*. Traditional LLMOps frameworks, such as *MLFlow* has also added some of these functionalities. These tools and frameworks help to continuously monitor the LLM pipelines and models for various metrics, such as token counts, latency, and throughput to keep a tab on various non‑functional requirements. These tools can also analyze the drift of key matrix and display them in intuitive dashboards. Some of these tools also have the functionality to detect real‑time anomalies and send out alerts to relevant stakeholders. Overall, I would highly recommend putting various monitoring and observability tools and mechanisms in place right from the start to ensure the reliability of the LLM and to get the most out of your LLM investment. So now that you have the LLM model selected and finetuned for your task at hand and you have put a monitoring layer in place, we will focus on integrating the LLM model to build the LLM‑powered application and integrate these LLMs into the wider enterprise ecosystem.

Integrating LLMs in Enterprise

LLMs as APIs

With the large language model blueprint selected and finetuned for the task, it’s essential to integrate these LLMs into enterprise applications and ecosystem seamlessly to create a competitive edge. Extending our Carved Rock Fitness company example, which has identified multiple LLM use cases from personalized emails to customer chatbots to analyzing customer feedback data. To leverage the power of LLMs, the team needs to integrate the LLM powered applications into the company enterprise ecosystem that are mostly available as multiple microservices for different tasks. To accomplish this goal, the team can also expose their LLM model as a microservice or API. This process is also known as **LLM Serving**.

Now, let’s see LLM Serving in action in a quick demo. Here in a Google colab notebook backed by T4GPU, I first installed the popular vllm library. Then, I’m also printing the machine IP, then running the LLM server for an open source, instruct tuned 7 billion Mistral model using the activation‑aware weight quantized, or AWQ version to reduce the memory requirement. The server is OpenAI compliant and can be used as a drop‑in replacement for your existing OpenAI API. The LLM model will be exposed as an API on port 8888, and I am using a proxy tunnel to access it from outside. Once running, you will have to open the proxy tunnel URL and paste the machine IP as a password for the first time, and then set the BASE\_URL and use the completion endpoint to provide the prompt and to generate a personalized email template. Switching back, if you have created multiple finetune models for different tasks, you may have created smaller LoRA adapters. One option for deploying these adapters is to have a dedicated deployment for each adapter. However, this approach may be costly if you have too many adapters. Consider using the dynamic adapter pattern to optimize cost. This involves swapping adapters on the same GPU hardware based on the incoming request. There are multiple LLM serving frameworks available. At the time of recording, some of the popular ones are TGI, or Text Generation Interface by HuggingFace, vLLM, OpenLLM, and Ray serve. LoRAX is a popular option especially for serving dynamic LoRA adapters. You can choose the right LLM serving framework aligned with your use case performance requirements, such as throughput and latency using the framework functionality of effective memory management and batching or using the distributed inference. Ease of setup across different environments, such as running as container on Cloud or on the Edge or local can be another consideration. You should also look at the flexibility of these frameworks across different types of settings, such as REST or gRPC endpoints or batching and streaming inference support or OpenAI compatibility for drop‑in replacement or support for finetune adapters. Documentation and community support could be important factor as well to pick the right serving framework.

LLM Integration with Enterprise Ecosystems

So far, we have learned that LLMs can be exposed as APIs to connect with other enterprise microservices, but if you have multiple LLM APIs for different tasks, you may also need an orchestration layer to connect them in a specific way. Popular frameworks for this are LangChain and DSPy, and if you’re looking for advanced orchestrations or agents, then you can also check out Langgraph, Autogen, or CrewAI, but to make the workflow more performant and cost effective, you should also consider putting a caching layer to cache the queries and responses to avoid going to LLM APIs for similar queries again and again, and there are popular frameworks like GPTCache and Redis cache if you are interested. Moreover, in an enterprise setup, upstream, structured, and unstructured data sources need to be connected, parsed, and ingested into vector databases, especially for RAG use cases, and as there are multiple moving parts in this ecosystem, it’s important to have a monitoring and observability layer for proper execution.

UI Design Considerations for LLM-powered Applications

Digital user experience has evolved with every technological shift. For example, earlier internet pages used to have scattered information on their web pages, and Google’s clean interface revolutionized this search. However, users still had to sift through top links for relevant information, and now generative AI has completely revamped this user experience. One of the reason for ChatGPTs popularity is its clean and powerful user experience. Users can input their questions in the search box, and the AI generates a response trained on the vast internet data without users having to search for information manually. There are additional options for users to regenerate responses or give feedback. Taking lessons from some of the popular LLM applications, let’s look at some of the important decision considerations for user interface starting with visual cues like magic wand, light bulb, or sparkle icons to suggest the presence of generative AI elements to the user. Other visual cues could be to help users on the way to interact with the application. For example, tools like Notion AI use a taskbar with a hint to generate content. Canva tool also provide some hints for users to get started. Users can also access contextualized menu similar to tools like Grammarly to improve writing, change tone, or fix grammar. Also, LLMs can be slow to process and generate responses. To address this, consider using animations and streaming options to display incremental content similar to ChatGPT interface, and if there is no incremental streaming response, at least show cues for processing and waiting. Moreover in the world of LLM, consider providing filters in the LLM application that users can adjust to improve response quality and align with the personal voice or tone or need. Also, you can give users the option to select different LLMs for the same task and let them compare and contrast the responses. It is also essential to have trust indicators. Users must be aware that the LLM‑powered engine that they are interacting with has limitations and flaws. Allow the users to control the prompt or response, such as regenerating the response if they are not satisfied, or pausing the generation if it is taking too long, or adjusting the prompt at any point. And since responses can vary a lot, it is important to capture user feedback through your user interface that can be fed back to improve the model results. Another important trust consideration can be to provide traces, such as verified source links to help users understand that the responses are grounded in facts. In summary, we are seeing a revolution in the user experience with LLM and Gen AI, which is mostly command driven and reduces the visual and cognitive load to surface information on demand in a personalized fashion.

Making LLMs Secure

While LLMs are very powerful and have opened new exciting opportunities, new risks have also emerged. For example, prompt leaking. Where the user prompt gets leaked to the malicious user or attacker leading to the loss of sensitive information, competitive prompt, or even intellectual property. Prompt injection is another security threat where the user prompt gets sandwiched with the attacker’s instructions to ignore the prompt and to execute malicious prompt to gain undue advantage or data poisoning where raw data used for training or finetuning of the LLMs gets polluted by the attacker compromising the integrity of the model and associated response generation or jailbreak where an attacker can access LLM weights or code and manipulate them to execute malicious intentions. Attackers can also launch the dreaded denial‑of‑service attack on the LLM model leading to model response quality deterioration or even making it unavailable. Such attacks can result in a massive cost of running costly underlying infrastructure. Though I have covered only a few security risks in this course, I highly recommend checking out the OWASP top 10 for the LLM report, and it is paramount that the appropriate measures be taken to tackle these LLM security challenges. For example, you can limit the privileges of the LLM system to the least necessary for the functionality and use robust input validation to filter potential malicious inputs from untrusted sources. Segregating untrusted content from the user input and controlling interaction with the external plugin can also protect sensitive information. You need to maintain the trust boundaries between the LLM, data sources, and plugins. You should also invest in frameworks for proper data sanitization and scrubbing to protect the sensitive data. Along with monitoring and observing different components of the LLM solution, consider using the OWASP framework for your LLM applications which helps identify and mitigate risks with practical checklists and integration with organizational cyber security controls to reduce vulnerability to threats. Mitre Framework, on the other hand, takes a resilience first approach to mitigate adversarial attacks in LLM and AI world. It includes a collection of attacks and learnings based on the real‑world observations. To put things in practical terms, you need to have both input and output controls between the LLM application and the LLM itself. These controls have to be guarded as per the organizational security needs. You can explore Python frameworks such as LLM Guard to accomplish this goal.

LLM Compliance and Regulation

LLM‑powered applications can only be beneficial for an organization if it is built and used in a compliant way. The compliance has to be with the organizational policy adhering to the company data access and LLM application usage. LLM responses must be aligned with the company and the brand tone and vision, and it should be used in a way that protects company intellectual property be data or application. LLMs application should also comply with ethical guidelines and maintain the transparency to ensure accountability. It is crucial to ensure that the LLMs are aligned with human values, meaning they are helpful, harmless, and honest. Moreover, the systems must be well tested to minimize biases. Not only this, LLM systems must comply with various applicable regulations such as EU AI act, General Data Protection Regulation, or GDPR, California Consumer Privacy Act, or CCPA, and more. Failure to comply with these regulations can lead to hefty fines and damage the company’s reputation. An organization must implement various measures to handle compliance requirements, such as scanning input and output data for sensitive information like PII, or personally identifiable information, and processing or filtering them as per the relevant regulatory requirements and standards. Advanced approaches may involve using encryption techniques, such as homomorphic encryption to handle sensitive data. You should also put layers to analyze the input and output data sentiments to detect any toxic or discriminatory elements and act accordingly. It is important to have easy and flexible configurations across various stages of LLM application, considering the existence of multiple levers, and changing compliance requirements. And while LLMs can automate or augment compliance tasks, human oversight remains critical to make ethical and context‑specific decisions, and it is important to maintain transparency and satisfy auditors by implementing dashboards, logs, and audit trails from the beginning. Throughout this course, we have explored various monitoring, observability, and security frameworks. Further, to make it easier to establish rules and guidelines as per the compliance requirements, consider exploring frameworks like Nemo Guardrail by NVIDIA and Guardrails AI. Overall, as Elon Musk said earlier that gen AI is the most powerful tool for creativity and has the potential to unleash the new era of human innovations. It’s essential that we use this power responsibly for the greater benefit of organizations and humankind. And with that, it’s a wrap for this course. Best of luck on your LLM and Gen AI journey and share your success stories on the discussion forum or social media.

**Prompt Engineering**

1.Which technique can reduce hallucinations in a large language model's response?

 Incorrect -Applying overfitting to the trained model

 **Correct** -Reinforcement learning from human feedback

 Your choice: incorrect -Using fine-tuning before training the model

 Incorrect -Reinforcement learning from random prompts' responses

 Incorrect -I don't know yet.

2.You have relied on a website for five years, learning to maintain a lush green garden. You notice that the website, owned and copyrighted by company XYZ, lacks a chatbot feature. You want to scrape the website content to build a chatbot for yourself. What mandatory steps would you take before creating the chatbot?

 Incorrect -Buy a large language model subscription that you will be using to develop the chatbot.

 Incorrect -Arrange a meeting with one of their operations team members and convince them verbally to agree with your project.

 Your choice: incorrect -Survey the existing gardening-related chatbots and analyze their pricing tiers to position your chatbot in the market.

 **Correct** -Obtain written permission from the company outlining your intended use of their website content.

 Incorrect -I don't know yet.

3.What is a potential effect of using a poorly-crafted prompt with an AI model?

 Incorrect -It could cause an increase in processing speed of the model.

 Your choice: **correct** -It could result in irrelevant or low-quality responses.

 Incorrect -It could lead to an increase in the cost of the model.

 Incorrect -It could cause the AI model to stop working.

 Incorrect -I don't know yet.

4.You are developing a large language model by training it on archive papers related to cryptography. During the model validation, you pass a prompt, and the model responds with a fabricated source by referencing it in a research paper. What steps would you take to reduce improper citations?

 **Correct** -Implement a validation step to verify all sources and references provided.

 Incorrect -Create 1000+ automated prompts and use their responses to retrain the model.

 Incorrect -Retrain the model using different medical domain papers to expand its knowledge base

 Your choice: incorrect -Replace archive papers with peer-reviewed research papers to reduce such issues.

 Incorrect -I don't know yet.

5.What best describes a math and reasoning task appropriate for generative AI?

 **Correct** -Tasks that require recognizing patterns and solving logical problems.

 Your choice: incorrect -Tasks that are based on emotions and feelings.

 Incorrect -Tasks which only require algebra-level problem solving.

 Incorrect -Tasks which cannot be quantified or measured logically.

 Incorrect -I don't know yet.

6.You create a prompt to arrive at a one-word answer, either "caution" or "neglect," as follows:  
"The dog's owner has not trimmed its nails in months, and they have become sharp and overgrown."  
The tool responds with a long paragraph instead of a one-word response. What minimal change must you make to your approach to obtain the expected output?

 Incorrect -Insert the phrase "Use complex language in your response." at the end.

 Incorrect -Include the sentence "Provide a detailed explanation." at the end.

 Your choice: **correct** -Add the phrase "Respond with one word:" at the end of the prompt.

 Incorrect -Rephrase the prompt as a question, asking for the dog's condition.

 Incorrect -I don't know yet.

7.Which best describes expanding texts tasks in the context of generative AI?

 Incorrect -Tasks that limit AI's ability to expand on a given topic.

 Incorrect -Tasks that encourage AI to store large volumes of text data for future reference.

 Incorrect -Tasks that emphasize optimization of AI's data storage and operational speed.

 Your choice: **correct** -Tasks that require AI to create or expand upon a short text or idea into a more detailed and comprehensive piece.

 Incorrect -I don't know yet.

8.In the context of generative AI, how would you explain the best way to use code generation tasks?

 Incorrect -Primarily use them to create new, complex software applications from scratch.

 Your choice: **correct** -Utilize them to handle repetitive tasks that do not require high-level decision-making.

 Incorrect -Use them primarily to decode online security systems.

 Incorrect -Deploy them to execute high-level tasks that require distinct human judgment.

 Incorrect -I don't know yet.

9.What describes the role of max token limits and how they impact the efficacy of generative AI chatbots?

 Incorrect -They empower the chatbot to use extra CPU resources for faster response times.

 Your choice: **correct** -Max token limits constrain the length of a sequence a chatbot can handle in a single interaction.

 Incorrect -Max token limits allow the chatbot to generate more creative and diverse outputs.

 Incorrect -They increase the chatbot's ability to moderate user-generated content and facilitate conversations.

 Incorrect -I don't know yet.

10.What best describes the operation of a large language model?

 Your choice: **correct** -It receives input text and predicts the next word or phrase.

 Incorrect -It transforms two-dimensional graphics into three-dimensional graphics.

 Incorrect -It tests software applications for bugs and potential improvements.

 Incorrect -It sorts and organizes big data sets.

 Incorrect -I don't know yet.

11.You must create a prompt that provides comprehensive instructions on maintaining a healthy aquarium environment for a goldfish. The response must highlight insights about water quality, tank size, feeding schedules, and known diseases. How will you write the prompt?

 Incorrect -Describe specifics on water quality, tank size, feeding schedules, and known diseases for an aquarium fish situated in the Arctic zone.

 Incorrect -Describe specifics on water quality, tank size, feeding schedules, and known diseases for a goldfish in its natural habitat.

 Incorrect -Give me information on setting up an aquarium for a fish. The response must describe specifics on water quality, tank size, feeding schedules, and known diseases.

 Your choice: **correct** -Give me information on setting up an aquarium for a goldfish. Include specifics on water quality, tank size, feeding schedules, and known diseases.

 Incorrect -I don't know yet.

12.How many prompts does a large language model require at minimum to accomplish text generation tasks when practicing few-shot learning?

 Incorrect -Zero

 Your choice: **correct** -Minimum two

 Incorrect -Three

 Incorrect -Maximum two

 Incorrect -I don't know yet.

13.What is the key characteristic of the few-shot prompting technique?

 Incorrect -It is the fastest prompting technique to get to the desired output.

 Your choice: **correct** -It involves passing a small number of examples to the model.

 Incorrect -It utilizes knowledge from the dataset it was previously trained on.

 Incorrect -It does not require any examples to be given to the model.

 Incorrect -I don't know yet.

14.When adjusting the temperature parameter during prompting, which aspect of the model's performance are you influencing?

 Your choice: **correct** -Creativity and randomness of responses

 Incorrect -Speed of responses

 Incorrect -Tone of responses

 Incorrect -Accuracy of responses

 Incorrect -I don't know yet.

15. You set the temperature value to 0.1 and the max tokens value to 20. You must pass two prompts but their responses must not be duplicates of each other i.e. the majority of the responses must be distinct. Of the following prompt sets, which prompts will you write to understand gardening?

 Incorrect -Prompt 1: Do you do gardening?  
Prompt 2: Do you do farming?

 Your choice: **correct** -Prompt 1: How to do gardening?  
Prompt 2: Highlight the benefits of gardening.

 Incorrect -Prompt 1: Elaborate the benefits of gardening.  
Prompt 2: Why should I do gardening?

 Incorrect -Prompt 1: What is gardening?  
Prompt 2: Define gardening.

 Incorrect -I don't know yet.

16. Which task is best suited for generative AI when inferring sentiment?

 Incorrect -Image recognition

 Your choice: **correct** -Text summarization

 Incorrect -Linear regression analysis

 Incorrect -Multivariate calibration

 Incorrect -I don't know yet.

17. You have a paragraph on buildings as follows:

 "Buildings, the epitome of architectural expression, arise from a fusion of creativity, engineering precision, and societal demands. These meticulously designed and constructed structures redefine cityscapes with their distinct forms and functions. Seamlessly blending art and utility, buildings stand as the embodiment of human ingenuity."

 You must create a prompt that asks the artificial intelligence (AI) tool to summarize the paragraph into three numbered lists. As per the guidelines of OpenAI, how would you write an influential prompt to arrive at the numbered list output?

 Incorrect -Restructure the following paragraph into an unordered list of three.

Paragraph: """ Buildings, the epitome of .... """

 Your choice: incorrect -Buildings, the epitome of .... Create a numbered list of three points on each topic.

 Incorrect -Restructure the following paragraph into a list.

Paragraph:

###Buildings, the epitome of ...

 **Correct** -Restructure the following paragraph into three numbered lists

Paragraph: """ Buildings, the epitome of .... """

 Incorrect -I don't know yet.

18.You recently made changes to prompts used in a language translation chatbot and need to determine the effectiveness of the changes. Which approach would be appropriate to gain insight into subjective metrics related to the app?

 Incorrect -Conduct load testing on the app

 Incorrect -Compare prompt performance against industry benchmarks

 Incorrect -Analyze quantitative performance metrics such as accuracy and speed

 Your choice: **correct** -Analyze qualitative feedback from user surveys and interviews

 Incorrect -I don't know yet.

19.You're working with a large language model that has a small context window. What could be the probable impacts on the model's performance?

 Incorrect - The model will perform better as smaller context windows make the task easier.

 Incorrect - The model's performance will not be affected whether the context window is substantial or small.

 Incorrect -Smaller context windows will increase the computational complexity of the model.

 Your choice: **correc**t -The model might fail to grasp longer dependencies in the text, affecting its capability to predict accurately.

 Incorrect -I don't know yet.

20. As a prompt engineer, how would you guide an AI model to recognize patterns from multiple examples and generate outputs similar to those examples?

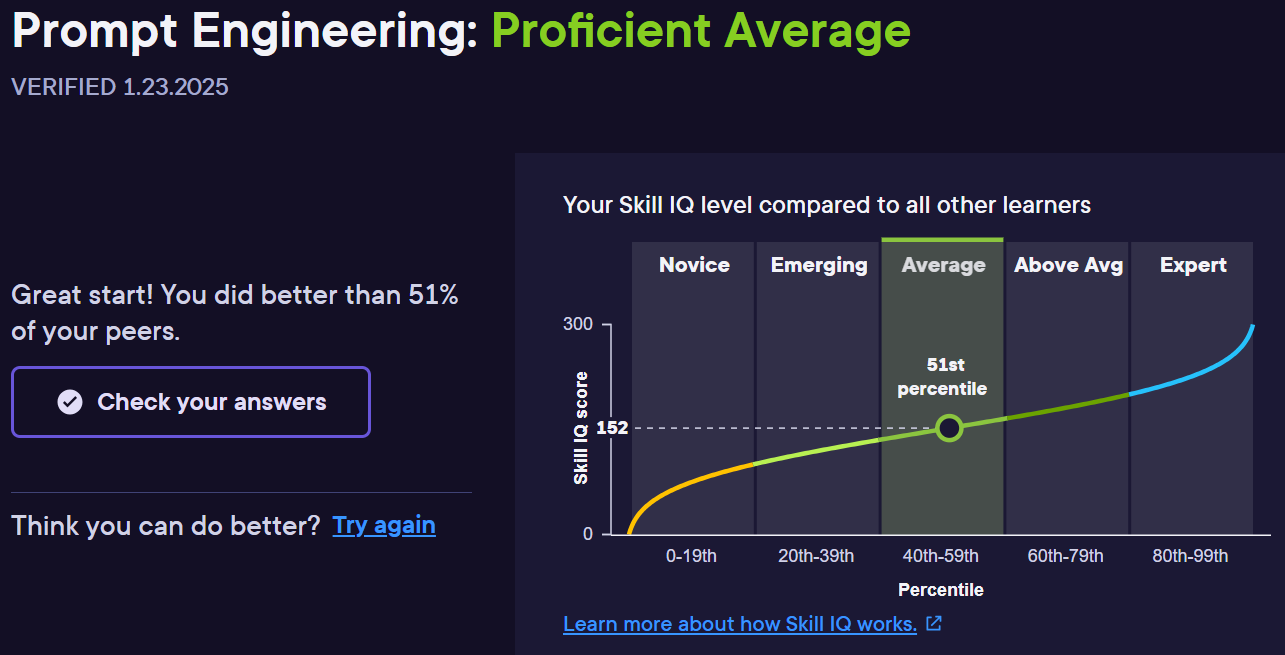
**** Your choice: **correct -By using few-shot prompting**

 Incorrect - By providing coherent instructions

 Incorrect -By providing clear and specific instructions

 Incorrect -By adjusting the temperature parameter

 Incorrect -I don't know yet.



As on 23-Jan-2025