Day - 1.1 : Foundational Models and Their Importance

Foundational models form the base of generative AI, built on large language model architectures like transformers. These models are pre-trained on vast datasets and are refined for specific tasks through techniques like fine-tuning and RLHF (Reinforcement Learning from Human Feedback). Google's Gemini models, for instance, are used to perform a range of tasks with high accuracy and efficiency, thanks to their large context window and multimodal capabilities.

Unique Characteristics of Gemini Models:

- **Extended Context Window:** Gemini 1.5 Pro can handle up to 2 million tokens, allowing it to process large documents or complex queries.
- **Multimodal Inputs:** Some Gemini models can process both text and visual data, making them versatile for tasks like image generation, analysis, and more.
- **OpenAI Compatibility:** Gemini's models are designed for easy integration with OpenAI SDKs, allowing developers to use them with minimal code changes.

Day - 1.2 : Prompt Engineering Techniques

Prompt engineering is a powerful tool to improve the performance and accuracy of large language models. By crafting precise prompts, we guide the model to produce desired responses.

Key Techniques:

1. **Zero-Shot Prompting**: Asking the model to perform a task without providing examples.
2. **Few-Shot Prompting**: Providing examples to guide the model's understanding of the task.
3. **Chain of Thought Prompting**: Encouraging the model to explain its reasoning in steps, which is particularly useful in complex calculations or logical tasks.
4. **Structured Output Modes**: Setting response formats like JSON to ensure output is organized for downstream applications.

Parameters in Prompt Engineering:

- **Temperature**: Controls the randomness in response generation. A higher temperature produces varied responses, while a lower temperature yields more consistent outputs.
- **Top-k and Top-p Sampling**: Techniques to limit the token selection pool, balancing creativity and coherence.

Day - 2 : Embedding

When someone enters a search query you convert that query into an embedding to and finally you find the web pages whose embeddings are the nearest neighbors to the query embedding in that vector space and that is basically how you get your search results.

classic metrics :- precision and recall

-> precision is all about how many of the retrieved items are actually relevant it is like are you hitting the target

-> recall tells you what proportion of all the relevant items you actually manage to find.

Precision at k and recall at k:

These are variations that focus specifically on the top k results.

-> higher **NDCG** means = better job of ranking.

NDCG is a metric used to evaluate the quality of ranking in information retrieval and recommendation systems.

-> RAG :-

RAG is all about using embeddings to make language models even smarter.

-> the basic idea is that you use embeddings to find relevant information from a knowledge base and then use that information to kind of boost the prompt that you give to your language model. This helps mode language models generate more accurate and relevant responses.

Types of embeddings:

i) Text embeddings :-

It allows you to represent words, sentences , paragraphs , even entire documents as these dense numerical vectors which is what makes them so powerful for all sorts of NLP tasks.

ii) Word embeddings :-

They are basically dense fixed length vectors that aim to capture the meaning of individual words.

(iii) document embeddings :-

are all about representing the meaning of longer chunks of text right like paragraphs or whole documents exactly and the white papers.

IV) image embeddings :-

image embedding as a method for computers to map local features from an image to a higher dimensional representation that is useful for image retrieval.

V) graph embeddings :-

graph embeddings all about representing objects and their relationships within a network.

Day - 3 : Generative AI agent :-

Generative AI agent is essentially a program that goes beyond what a standalone generative AI model can do; it adds in reasoning logic and crucially access to information outside of its internal knowledge.

-> it's not creating text or images, it's actually trying to achieve a specific goal. and it is observing the world around it and then taking actions based on what it sees using whatever tools it has available to it.

-> language model is the brain of the operation.

-> prompt engineering also plays an important role here.

-> prompt engineering is all about crafting those really effective instructions for language models and that's crucial for guiding the agents throughout the process.

-> ReAct: Reason and action.

It is a framework where you encourage the language model to think step by step to reason and then to interact with the environment using tools that are the Act part based on what the user is asking.

-> COT (Chain of Thought) :-

COT is all about getting the model to think through a problem in a very structured way like writing out a series of logical steps that lead to the solution instead of just asking for the answer directly you guide the model to explain its thought process step by step.

-> TOT (Tree of Thought) :-

It is a kind of kale on steroids. It's designed for problems where you need to explore lots of different possibilities to be more strategic instead of just a single chain of thought. It's like branching out exploring multiple paths.

-> Flight API :-

its use to interact with an external tool to get the information it needs.

Tools are breakdown in three main categories :-

-> extensions

-> functions

-> data stores

1) extensions :-

" are essentially a standardized way to connect an agent to an API. They make it really easy for the agent to use the API without having to worry about all the low-level details of how that API actually works.

2) functions :-

functions are kind of like functions in regular programming they're self-contained pieces of code that do a specific thing but in the agent world the language model decides when to use each function and it figures out what arguments to give it based on the function specification and the.

3) data stores :-

data stores used to store and search.

-> example - vector database.

Day - 4 : Solving Domain specific problem using LLM :-

Large Language Models (LLMs) are demonstrating increasing potential in solving domain-specific problems, moving beyond general-purpose tasks. However, effective application requires careful consideration of the specific domain and strategic prompting.

Strategies for Effective Domain-Specific LLM Usage:

- **Prompt Engineering:**
  - Crafting clear, specific, and context-rich prompts is essential.
  - Use role prompting to simulate domain experts.
  - Employ chain-of-thought prompting to guide step-by-step reasoning.
  - Provide examples and constraints to shape the LLM's output.
- **Retrieval-Augmented Generation (RAG):**
  - Integrate external domain knowledge bases into the LLM's workflow.
  - Retrieve relevant documents and information based on the user's query.
  - Use the retrieved information to augment the LLM's response.
- **Fine-Tuning:**
  - Adapt the LLM to the specific domain by fine-tuning it on relevant datasets.
  - This can improve the model's accuracy and performance on domain-specific tasks.
- **Tool Augmentation:**
  - Allow the LLM to use external tools, such as code interpreters, or API's, to increase the accuracy of the responses. For example, if the domain is finance, allowing the LLM to use a calculator, or a finance API, would improve accuracy.
- **Hybrid Approaches:**
  - Combine LLMs with traditional domain-specific methods (e.g., rule-based systems, statistical models).
  - Use LLMs to augment or enhance existing workflows.

Day - 5 : MLOps for Generative AI

MLOps for Generative AI adapts traditional machine learning operations to the unique challenges of generative models. Key aspects include:

- **Data Management:**
  - Handling diverse, often unstructured data for training.
  - Ensuring data quality and provenance for generated content.
- **Model Development:**
  - Experimenting with various model architectures and parameters.
  - Efficiently fine-tuning and adapting models for specific tasks.

- **Model Deployment:**
    - Deploying and scaling generative models for real-time inference.
    - Managing resource-intensive computations.
- **Monitoring & Evaluation:**
    - Evaluating generated content for quality, relevance, and safety.
    - Monitoring model performance and detecting drift.
    - Implementing human feedback loops.
- **Safety & Responsible AI:**
    - Addressing biases, toxicity, and potential misuse of generated content.
    - Implementing safeguards and content moderation.
- **Version control:**
    - Tracking Prompts, model weights, and datasets.
- **Automation:**
    - Automating pipelines for training, evaluation, and deployment.