Contents

1 Fundamentals 2

2 Solution Development 3

3 Application Development 5

4 Application Evaluation and Governance 6

5 Application Deployment and Monitoring 7

1 Fundamentals

- LLM: Model trained on massive datasets to achieve advanced language processing capabilities
- Foundation Models: Large ML model trained on vast amount of data & fine-tuned for more specific language understanding and generation tasks

Components:

- Encoder: Converts text input into tokens (embeddings)
- Decoder: Converts generated output tokens back into meaningful words
- Transformer: Train the token embeddings

Databricks AI:

- GenAI (Custom models, model serving, RAG)
- End-to-end AI (MLOps with MLFlow, AutoML, Monitoring, Governance)
- Databricks + MosaicML:
  - Rapid democratization of model capabilities
  - Making GenAI models work for enterprises
  - Unifying AI and data stack
  - Advantages: Customize models, secure environment, competitive

Legal and Ethical Considerations:

- Prompt Injection: Inserting a specific instruction or prompt within the input text to manipulate the normal behavior of LLMs
- Prompt Engineering: Designing and crafting effective prompts or instructions for generating desired outputs from an LLM

2 Solution Development

From Prompt Engineering to RAG:

- Prompt: Input or query given to a LLM to elicit a specific response
- Prompt engineering: Practice of designing and refining prompts to optimize the responses generated by an AI model
- Prompt components: Instruction, context, input/question, output type/format
- Prompt techniques:
  - Zero-shot: Prompt that generates text or performs a task without providing any examples or additional training specified to that task
  - Few-shot: Prompt provides with a few input-output examples to guide the model for generating the desired output
- Prompt chaining: Multiple tasks are linked together, with the output of one prompt serving as the input for the next → allows for more complex tasks to be broken down into manageable steps
- Chain-of-thought (CoT) prompting: Enhances the reasoning capabilities of LLMs by guiding them to articulate their though processes step-by-step

Tips and tricks:

- Different models may require different prompts
- Provide examples and clues to guide model's response generation
- Different use cases may require different prompts
- Use delimiters to distinguish between instruction and context
- Ask the model to return structured output
- Ask the model not to hallucinate, not to assume, not to rush

RAG:

- Passing context as model inputs improves factual recall
- Retrieve data/documents relevant to a question/task, provide them as context to augment the prompts to an LLM to improve generation
- Components:
  - Index & embed: Embedding model creates vector representations of the documents and the user query
  - Vector store: Specialized to store unstructured data indexed by vectors
  - Retrieval: Search stored vectors using similarity search to retrieve relevant information
  - Filtering & Reranking: Process of selecting or ranking retrieved documents before passing as context
  - Prompt augmentation: Prompt engineering workflow to enhance context via injection of data retrieved from the vector store
  - Generation
- Flow:
  - Documents are embedded
  - User asks question and LLM converts query to embeddings
  - Similarity search for embedded documents and embedded user query
  - Similar documents are passed as context to LLM generation model
  - Model augments and generates completion
- Databricks:
  - DLT

  - Mosaic AI Model Serving (embeddings, foundation models, custom models)
  - Mosaic AI Vector Search
  - Unity Catalog's Model registry
- Benefits:
  - Up-to-date and accurate responses,
  - Reducing inaccurate responses or hallucinations
  - Domain-specific contextualization
  - Efficiency and cost-effectiveness

Preparing Data for RAG Solution:

- Issues:
  - Poor quality model output
  - Lost in the middle paradigm
  - Inefficient retrieval
  - Wring embedding model
- Data prep process:
  - Ingestion → Data storage → Chunking → Persist in vector store
  - Delta Lake for storing data and tables in the Databricks DI Platform
  - Unity Catalog for governance, discovery, access control for RAG applications
  - Document extraction: Split documents into chunks, embed chunks with a model, store them in a vector store
- Chunking data:
  - Context-aware chunking: By sentence/paragraph/section, leverage special punctuation, include metadata/tags/titles
  - Fixed-size chunking: Divide by a specific number of tokens, simple and computationally cheap method
  - Chunking overlap defines the amount of overlap between consecutive chunks ensuring that no contextual information is lost
  - Window summarization: Each chunk includes a windowed summary of previous chunks
- Embedding model:
  -

3 Application Development

4 Application Evaluation and Governance

5 Application Deployment and Monitoring