LLM CONSIDERATIONS

**Prerequisites**

* API key to Foundational Model(s)

Which GenAI framework to use? (Langchain, CrewAI, ..)

Which foundational model family to use? (text to image, text to text etc.)

Which foundational model to use? (GPT4,Llama..)

* How recent is the data it was trained on?
* Cost per token
* Context window limit

Which architecture/design pattern to use? (RAG, multi-agent-system (MAS), )

Which prompts to use ?

* System prompt - sets context (role,behaviour,goal), used in all questions. Ensures consistency & alignment
* Human Prompt - what user asks, comes after system prompt
* Combined Prompt - system prompt + human prompt + optional other elements (history / context…)

How to handle complex queries? Restrict to single one?

How to handle memory ?

How to evaluate & monitor solutions?

**Assuming a RAG system, following configs to tune:**

* Which documents / how many to include ?
* How to chunk document ? e.g. chunk size & overlap
* Which embedding model to use?
* Retrieval parameters
  + No. of embeddings to retrieve from VectorStore ? Should criteria be top K or similarity score threshold?
  + How to combine retrieved embeddings (e.g. 'stuff' concats)

**Costing**

* What cost plan does Nestle have in place ?
  + Is it per token or is there a flat fee ??
  + If per token, need to find a way to visualize the cost impact so users know how much they are actually using for every invokation of the model
* Limit access to API keys

**Evaluation**



**Privacy**

* How to avoid PII data being stored and thus appearing in potential answers

Why GenAI ?

* Super exciting new field
* Move fast & be responsible mentality
* Ideation to deployment in a fraction of the time so able to test & learn so quickly and unlock more value
  + Used to be months using traditional supervised learning approach
    - Get labelled data -> Train & tune model -> Deploy
    - Had to choose one strategy, now can develop 20 and see which sticks
  + LLM Based development
    - Define problem & choose foundational model -> define prompts -> deploy

Biggest challenges

* Tuning the config for ai systems (chunk size, chunking strategy .. )
* Evaluating these models (bit of a bottleneck?)
* LLMOps
* Software integration
* Bias in models
* Ensuring guardrails
* Avoiding hallucinations
* Limiting toxicity

|  |  |
| --- | --- |
|  |  |
| embedding | mathematical representation of the semantic meaning of a word, sentence, or document. projection of a concept in a high-dimensional space. Where related concepts will be close to each other, while concepts with different meanings will lie far away.  Use    <https://dashboard.cohere.com/playground/embed> to visualize embeddings in two dimensions. |
| Prompt engineering | art of crafting clear and concise instructions for the LLM to achieve the desired outcome |
| Prompt Templates |  |
| Transformers |  |
| RAG |  |
| Vector Database |  |
| Shot learning | how much instruction an LLM needs to learn a new task.   * **Zero-Shot:** No examples provided, the LLM relies on its existing knowledge. * **One-Shot:** Just one example is given to guide the LLM. * **N-Shot:** Multiple examples are provided for the LLM to learn from. |
| Mixture of Experts (MoE) |  |
| Indexing |  |
| LLM Agents / Agentic workflow | Rather than just ask the model to do task A (a non-agentic workflow i.e. zero-shot), you introduce planning, e.g. split it into subtasks, ask it to evaluate/revise each, retain memory --> improved answer (feedback loop)   * Introduces reasoning * Might be slower, but is better * Necessary for very complex queries * E.G. "In light of new data privacy laws, what are the common legal challenges companies face, and how have courts addressed these issues?   + access legal databases to retrieve the latest laws and regulations.   + Then establish a historical baseline of how similar issues were previously handled   + Summarise legal documents   + Forecast future trends based on observed patterns   + i.e. needs a reliable memory to track progress, and access to necessary tools   Agents often consist of 4 components:   1. Agent/brain (foundational model) 2. Planning    * CoT method (chain of thought) -> adaptive strategy where agents tackle sub-tasks one by one    * Tree of Thought (ToT) method -> exploring different paths to solve a problem. Generates multiple ideas at each step and explores them like branches on a tree    * ReAct & Reflexion are two approaches to incorporate feedback 3. Memory 4. Tool use |
| Coder agent |  |
| Vector DBs | Pinecone, redis, mySQL |
| LangGraph |  |
| LangSmith |  |
| Mem0 / MemGPT |  |
| Dotenv | Package to import API keys hosted in .env file to instantiate foundational models / vector DBs |
| Gen AI Frameworks | Langchain, Crew AI, AutoGen  Each has own strength, but langchain has large community, easier learning curve, gr8 integration (with tensorflow/pytorch/hugging face etc.), but maybe not as customisable / flexible |
| Context Window | includes prompt templates + instructions + user input + text generated by model  If a model has a context window of 2048 tokens and 200 tokens are reserved for a prompt template, only 1848 tokens remain for user input and generated output  If ask 'Where should I go in holiday next year?' -> Cambodia -> 'what is the temperature in june there?' (it knows what you are referring to  Can hallucinate / repeat once context window exceeded (as truncates initial part of convo)    Strategies to mitigate   * Summarise a chunk of previous conversation every now and then to avoid exceeding limit * Ask questions sequentially |
| Chunking approaches | * + CharacterTextSplitter - fixed length tokens   + SentenceSplitter   + Document based chunking i.e. based on inherent format of doc   + PythonCodeTextSplitter - by functions/class   + Agentic chunking whereby LLM determines chunks (expensive)   + Semantic Chunking - e.g. Llamaindex semantically picks breakpoints between sentences using similarity |

**#### Handling complex/ambiguous user queries? ####**

* Translate into a form more suitable for the LLM
* Multi-query
* RAG fusion
* Decomposition
* Step Back
* HyDe
* Recursive
  + Breaks complex question into multiple questions. Q1. fed to LLM to get Ans1. Q2 & Ans1 fed to LLM2 etc. Will then have a final layer that considers answer of each to feed into LLM for final ans.

**### Agentic AI Workflows/Reasoning Design Patterns ###**

1. Reflection
   * Self-reflection i.e. Feed answers from LLM back as input, can optimise or even correct own code
   * Reflection with LLMs - feed answer from one LLM into another foundational model
2. Tool Use
   * API calls
   * Tools can be email, google calendar, wikipedia, search engine, code execution tool, a recommender system developed by your DS team, a database
   * E.G. What is the best coffee maker in UK -> api call to copilot for websearch
3. Planning
   * Creating steps
   * Might chain multiple foundational models together to arrive at final solution
   * E.G. If want DB of runners faces and bib numbers, might split into numbered steps. 1. detect face using bounding box 2. detect bib using text & bounding box 3. find face closest to bib based on euclidean distance 4. add record with face and bib no. to DB 5. iterate through steps 1-4 until for all frames in a video
4. Multi-agent collaboration (MAS - multi agentic systems)
   * Different agents prompted at different times
   * Potentially mixture of experts ?
   * Andrew Ng says future is MAS
     + LLM specialists rather than an LLM generalist
     + One LLM great at configuring computers, one at writing emails, another great at coding, one at writing poems. Each trained on very specific data sets. Means lots of smaller speciailised models can do what a single large LLM can do better. A LLM will then decide which of these specialised LLM models to use

**GenAI tech stack - Tools to build a Gen AI Application**

* Layer 0 - semi-conductors
  + Nvidia / intel / AMD
* Layer 1 - Cloud infrastructure
  + Aws / gcp / azure / snowflake
* Layer 2 - Foundational models
  + openAI / anthropic / meta's llama
* Layer 3 - Applications (Mainly Frontend)
  + Common sense / workera / speechlab etc

Reference Architectures for LLMs

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Description** | **Best For** |
| RAG | A RAG is not an AI agent  Not autonomous  Cannot solve complex problems. They just retrieve proprietary/additional data  RAG can be part of an AI agent |  |
| Fine-tuning | Adjusting parameters in LLM transformer model for custom dataset |  |
| Tool Augmented |  |  |

**Comparison of Architectures**

|  |  |  |  |
| --- | --- | --- | --- |
| **Architecture** | **Best For** | **Complexity** | **Drawback** |
| **RAG** | Knowledge-intensive tasks | Medium | Requires vector database and embeddings. |
| **Tool-Augmented** | Real-time or calculation-based queries | Medium | Depends on tool integration. |
| **Knowledge-Grounded** | Fixed domain knowledge | Low | Static and lacks dynamic updates. |
| **Memory-Augmented** | Long-term interaction or adaptation | High | Memory management can be challenging. |
| **Agent-Based** | Multi-step or goal-oriented tasks | High | Task decomposition is complex. |
| **Multi-Modal** | Image, text, or video input/output | High | Requires advanced model integration. |
| **CoT Enhanced** | Complex reasoning | Medium | Not always intuitive for all users/tasks. |

Custom Chains

* Student wants to study for GCSEs, you upload physics, maths, german TB to VectorDB

**Evaluation**

BLEU, ROUGE, and perplexity. Additionally, I use human evaluation for assessing coherence and relevance. For image-based models, I rely on FID (Fréchet Inception Distance) and IS (Inception Score) to measure visual fidelity and diversity. When applicable, I also run application-specific tests, such as assessing how well generated data improves downstream tasks."

**Situational Questions**

* How would you build a chatbot for your company on nestle products?
* How would you build a natural language to SQL application ?

Ai Agents

Airbnb agent not a chatbot since it has access to tools

LangChain

* enabled new wave of AI Applications
* Allows you to chain LLMs together / tasks together of otherwise high complexity

Multi Agentic Systems

AI Agents vs AI Automations

Conversational chatbots

Voice agents

* Access to three tools (google calendar,

EXAMPLE WORKFLOWS

RAG

* Upload document --> Chunk doc --> Embedding --> Store in VectorDB & Index --> Provide Prompt Template
  + Human Query -> Embeds question & Similarity Search against VectorDB --> Returns top K embeddings for most relevant information --> feeds this & original question to LLM --> Answer returned

MAS - E.G. Show me Apple stock for last 6 months

* Connect to web API to get current date - figures out this 6 month period
* Uses this data to search using search API
* Generates graph using another API

Prompt engineering

<https://platform.openai.com/docs/guides/prompt-engineering>

1. Include details in your query to get more relevant answers
2. Ask the model to adopt a persona
3. Use delimiters to clearly indicate distinct parts of the input
   * In system prompt would tell it how to expert user input
   * So would have to tell user how to use tool optimally for best ans
4. Specify the steps required to complete a task
5. Provide examples
6. Specify the desired length of the output
7. Allow the model time to think

e.g. in system prompt: First work out your own solution to the problem. Then compare your solution to the student's solution and evaluate if the student's solution is correct or not. Don't decide if the student's solution is correct until you have done the problem yourself.

GENAI DESIGN PATTERNS

<https://towardsdatascience.com/generative-ai-design-patterns-a-comprehensive-guide-41425a40d7d0>

1. Layered Caching Strategy Leading To Fine-Tuning
2. Multiplexing AI Agents For A Panel Of Experts
3. Fine-Tuning LLM’s For Multiple Tasks
4. Blending Rules Based & Generative
5. Utilizing Knowledge Graphs with LLM’s
6. Swarm Of Generative AI Agents
7. Modular Monolith LLM Approach With Composability
8. Approach To Memory Cognition For LLM’s
9. Red & Blue Team Dual-Model Evaluation

#### deepchecks webinar notes

compare different agents having made some changes to prompts / documents included / token vector /

Can integrate custom metrics as part of dashboard experience

* + Could be input metrics e.g. no. of email addresses included in document input
  + Could be output metrics e.g. text length from LLM output / compression ratio /

* + No test set in generative AI
  + Add logging steps
    - Adds questions multiple times to different chatbots - one with higher chunk overlap vs different chunk size
    - Each chatbot will log into a different version of deepchecks

* + VALIDATES PROB OF HALLUCINATION

Can create metrics such as below

* + And could create your own thresholds for each of these
  + If AI agent does not surpass threshold then maybe wont consider

**Toxicity**

A measure of how harmful or offensive a text sample is (0 to 1), uses the SkolkovoInstitute/roberta\_toxicity\_classifier model N/A may indicate ongoing calculation or property-specific conditions.

**Grounded in Context**

A score between 0 and 1, where 1 means that the LLM output is fully based on the input context, and 0 means that the LLM output is not based on the input context at all.

**Objectivity**

Evaluates the presence of gender, racial, or political bias in the LLM output. A high score indicates output free from detectable bias, promoting fair and inclusive communication.

**Text Quality**

Measures the overall linguistic excellence of the output, combining grammatical accuracy with coherent expression of ideas. A high score indicates text that is not only technically correct but also flows logically and reads naturally, enhancing overall effectiveness.

**Completeness**

Addressing all parts of the request. The output should provide a full and complete solution to all the parts of the request. The output does not require follow up questions in order to fully solve the original request. Annotators were instructed to penalize outputs which ignore parts of the request or avoid from giving an answer.

**Retrieval Relevance**

How relevant the retrieved documents (information\_retrieval) are to the user input, with a score ranging from 0 (not relevant) to 1 (very relevant)

**Avoided Answer**

The likelihood that the LLM avoided answering the input question (0 to 1).

**Sentiment**

Sentiment of the text, calculated using the TextBlob sentiment analysis model. Ranging from -1 (negative) to 1 (positive)

Other experiments

* + Change data sources
  + Change system template - to give shorter answers

#############################