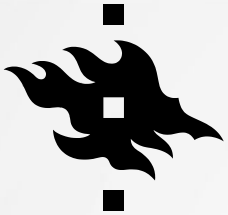


Introducing Prompt Engineering: What, Why and How?

Guanghan Wu

Department of Computer Science

Univ.Helsinki

What are Language Language Models (LLM)?

- Deep learning models pre-trained on vast amounts of textual data. E.g., GPT-4 has 1.76 trillion parameters.
- Mostly based on the transformer architecture, utilizing self-attention mechanisms to capture complex textual relationships.
- Able to understand and generate human-like text, capable of various tasks like translation, summarization, code generation, question-answering, etc.

What are Language Language Models (LLM)?

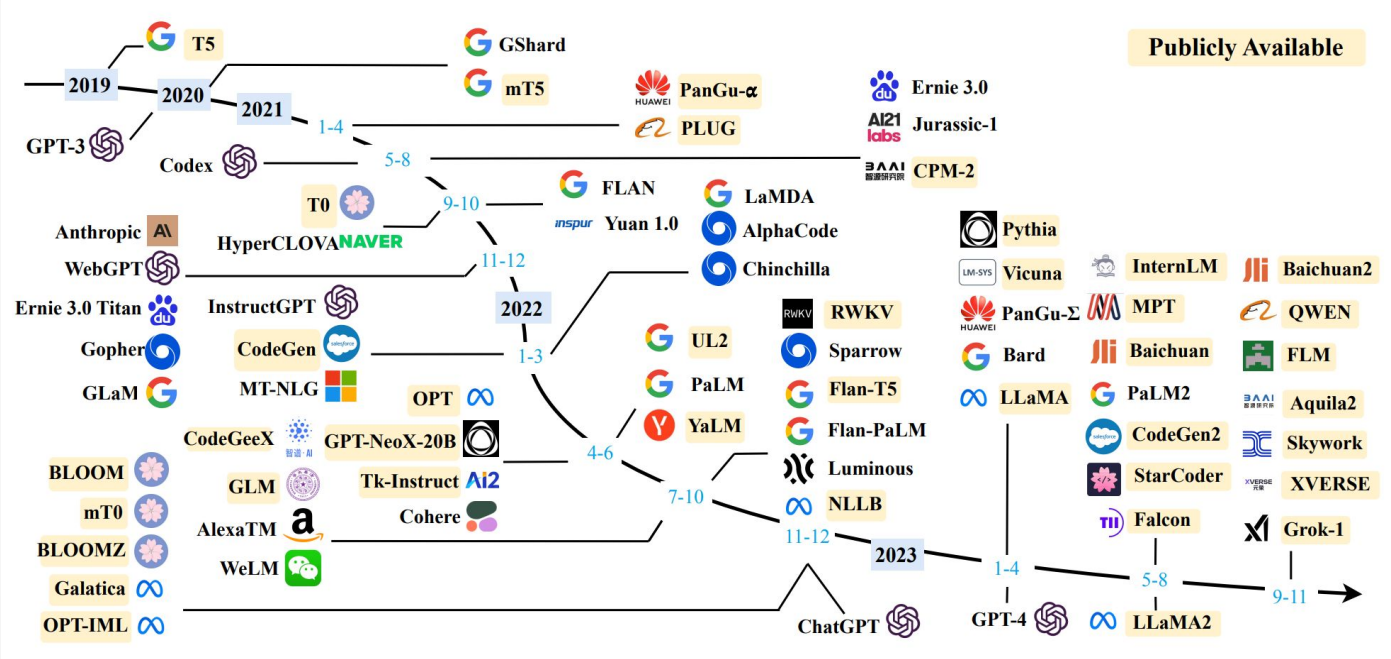


Fig 1. A timeline of existing large language models (having a size larger than 10B) in recent years. [4]

What Can LLMs Do?

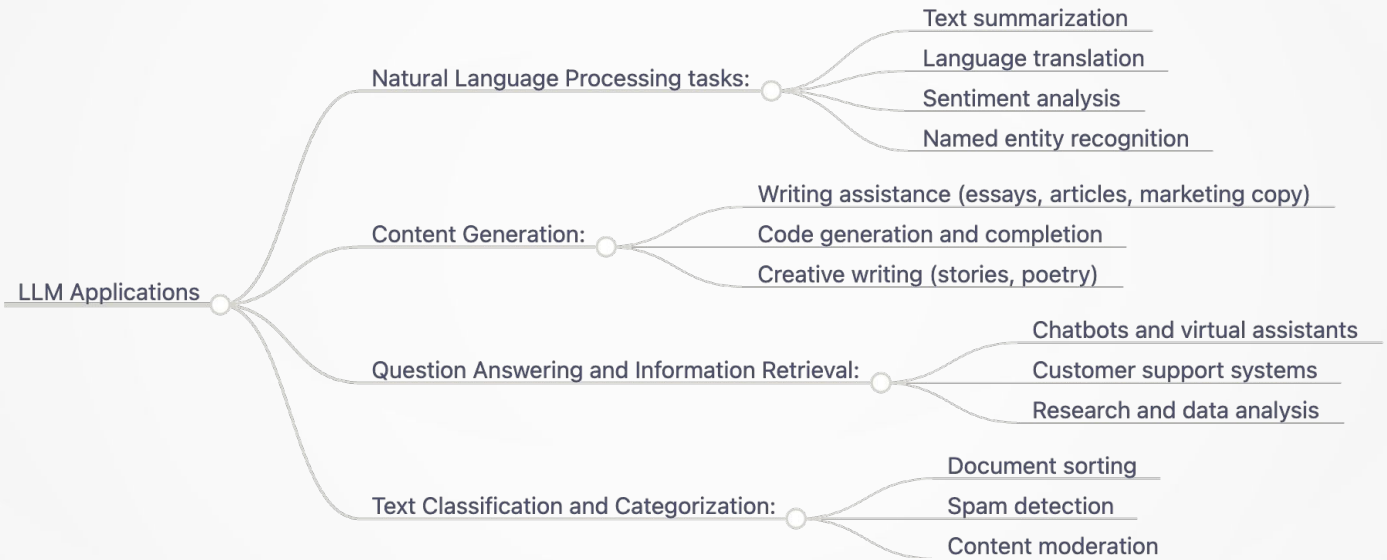


Fig 2. The applications of LLMs in various domains. Adapted from [5]

What is Prompt Engineering?

Essence: Crafting the optimal prompt to achieve a specific goal with a language model.

- Domain knowledge, understanding of the AI model, and a methodical approach to tailor prompts for different contexts.
- Construction of prompts.
  - Creating templates that can be programmatically modified based on a given dataset or context, i.e., prompt templates.
- An iterative and exploratory process,
  - akin to traditional software engineering practices such as version control and regression testing.

Why Prompt Engineering?

Conclusion:

Advanced prompt engineering and specialized techniques are needed to enhance LLM utility and mitigate inherent constraints.

LLMs have their limitations:

- Transient State:
  - Lack persistent memory or state, need additional systems for context retention and management.
- Probabilistic Nature:
  - Different answers even with the same prompt.
- Outdated Information:
  - Pre-trained on historical knowledge, precluding real-time updates.
- Content Fabrication:
  - Hallucination: LLMs may generate plausible yet factually incorrect information.
- Resource Intensity:
  - The substantial size of LLMs means huge costs, impacting scalability and accessibility.
- Domain Specificity:
  - Domain-specific data often required for specialized tasks.

Prompts: Benefits & Limitations

Benefits

- Describe tasks precisely & creatively
- No expert skills needed (Configurations; Code)
- Multi-lingual inputs
- ...many more

Limitations

- Limited no. words per prompt
- Limited prompt per chain
- Required domain knowledge
- Lack of transparency
- Lack of repeatability

Prompt Design Principles & Skills

Be concise.

- Brevity & clarity
- As simple as possible

Examples:

✖ *"Can you provide me with a detailed explanation of the photosynthetic process and the significance of this biochemical reaction?"*

✔ *"Explain the process of photosynthesis and its significance in detail."*

Be clear.

- Avoid vagueness & ambiguity
- As specific as possible

Examples:

✖ *"How do I produce a paper?"*

✔ *"Explain the common steps of writing a research paper for a peer reviewed academic journal.'*

Prompt Design Principles & Skills

Break down complex tasks.

Avoid…

- combining several tasks in one prompt
- asking for several outcomes/formats in one prompt

Divide tasks involving several steps or topics..

- into separate prompts
- or make it into a prompt chain (= conversation)

Example:

“Design a comprehensive marketing campaign for a new eco-friendly product, detailing the target audience,key messaging, and promotional strategies. include specific examples of advertising mediums such as social media, print, and television, and create a visual mockup of an advertisement in a modern minimalist style.Write meaningful alt-text for the advertisement, summarizing its content for enhanced accessibility. Finally draft a 2-minute pitch script for presenting this marketing campaign to potential investors.”

Prompt Design Principles & Skills

Include context & logical structure.

- Provide context
- Structured & coherent prompts

Examples:

✖ *"Mention tasks involved in writing a research paper."*

✔ *"List the steps involved in writing a research paper for a student assignment. Begin with selecting a topic, end with proofreading the final draft."*

Specify desired output.

- Style & tone
- Depth & length
- Format, language, or content type
- Temperature (parameter controlling precision & creativity

Examples:

Format, language, content type:
"For example: A three-column table; a poem in Swahili; a script in Python."

Temperature:
"Balance creativity & precision, depending on task"

Prompt Design Principles & Skills

Teaching in the prompt.

- LLMs an learn from what they are being fed in the prompt: zero-shot learning ability.
- Algorithms, knowledge, functions…

Chain of thought prompting.

- Force it to follow a series of steps in its “reasoning”
- By adding “**let’s think step by step**” at the end of a prompt.
- Zero-shot Chain of Thought prompting

Being forceful.

- Do not be so nice and friendly.
- Use forceful language sometimes.
- ALL CAPS and exclamation mark work!

Reordering of the examples and the prompt.

- LLMs like GPT only read forward.
- Giving the instruction before the example helps, even the reordering of the examples works.

A Prompt Framework: CO-STAR

Context(C): Providing background information helps the LLM understand the specific scenario.
Objective(O): Clearly defining the task directs the LLM's focus.
Style(S): Specifying the desired writing style aligns the LLM response.
Tone(T): Setting the tone ensures the response resonates with the required sentiment.
Audience(A): Identifying the intended audience tailors the LLM's response to be targeted to an audience.
Response(R): Providing the response format, like text or json, ensures the LLM outputs, and help build pipelines.

Advanced Techniques in Prompt Engineering – Chain of Thought (CoT)

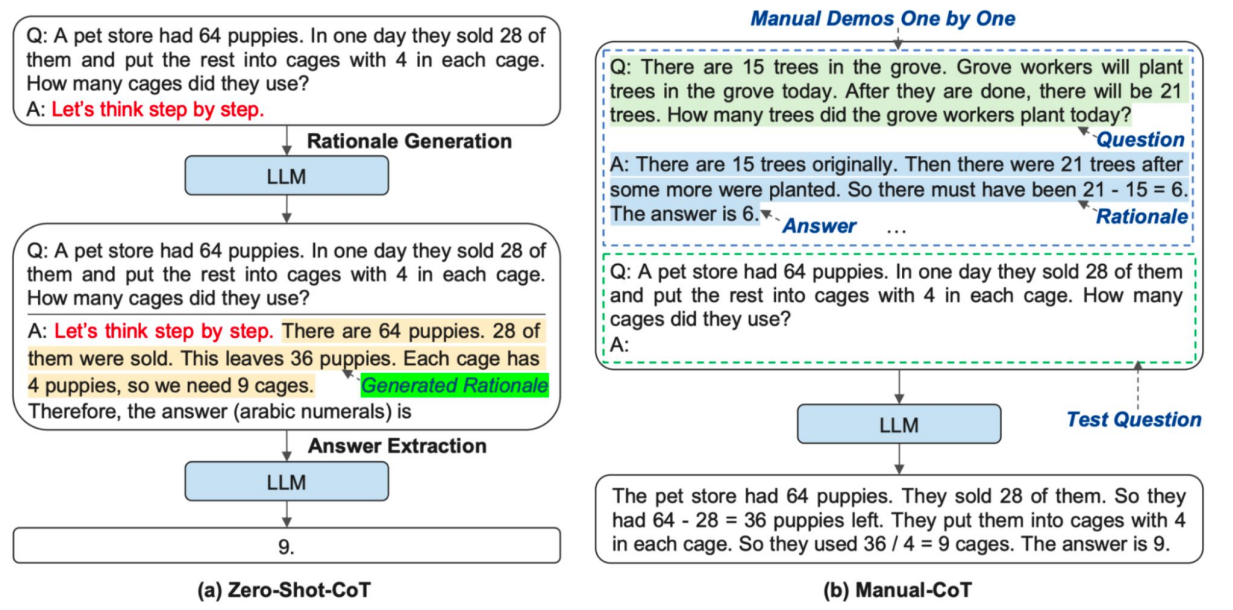


Fig 3. Comparison of Zero-shot and Manual Chain of Thought (few-shot) techniques. [2]

Advanced Techniques in Prompt Engineering – Tree of Thought (ToT)

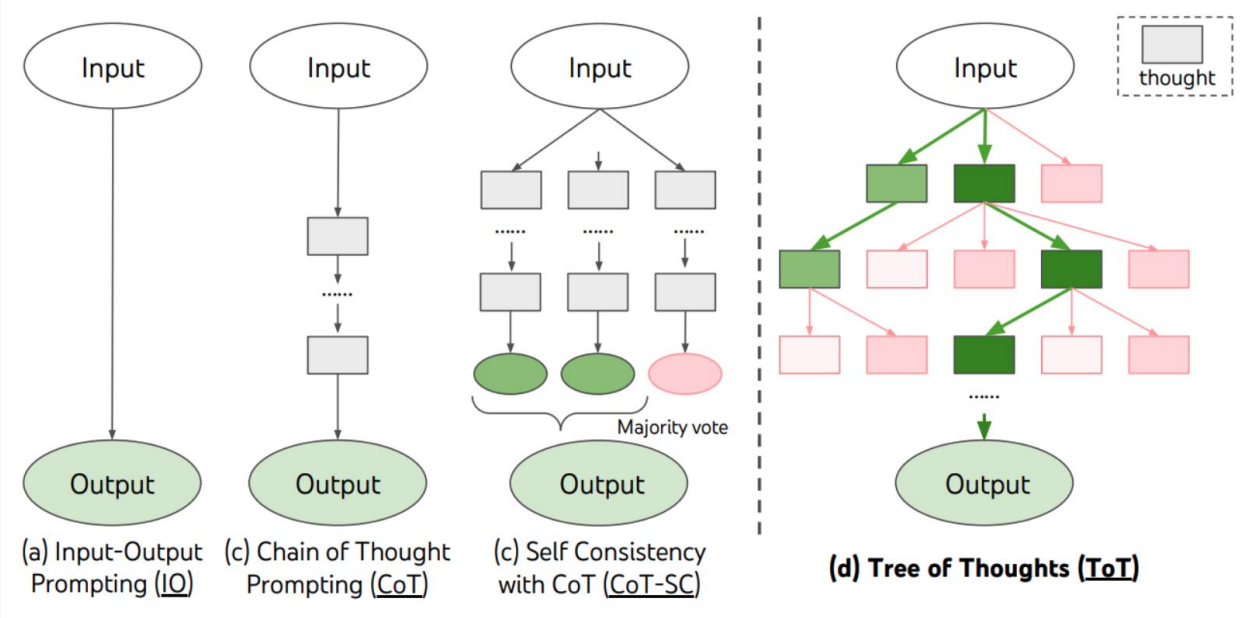


Fig 4. Illustrative representation of the Tree of Thought.. Each branch symbolizes a distinct line of reasoning, enabling a comprehensive exploration of potential solutions. [2]

Advanced Techniques in Prompt Engineering – Reflection

Reflection is a recent advancement in LLM technology that enables self-improvement through introspective evaluation.

Process: After generating an initial response, the LLM critically assesses it against predefined criteria. If issues are found, the model refines its response, potentially creating multiple improved versions.

Challenges: Accuracy depends on the LLM's training in self-evaluation; Risk of reinforcing errors if self-assessment is flawed

Implications: Reflection could lead to more autonomous, versatile, and reliable LLMs, especially for applications requiring high precision.

Advanced Techniques in Prompt Engineering – Streamlining Complex Tasks with Chains

Chains are a method for using LLMs in complex, multi-step tasks.

- Sequential linkage of specialized components to break down complex tasks.
- **Output from one component becomes input for the next**.
- Range from simple (information retrieval) to complex (reasoning, decision-making).
- Examples: Medical diagnosis chain (symptom collection → differential diagnosis → treatment recommendation).

Advanced Techniques in Prompt Engineering – Retrieval Augmented Generation (RAG)

“Mounting” an external knowledge database for LLMs to deal with its limited context window and outdated knowledge, improving accuracy and relevance of LLM responses.

Retrieval: The system searches a database for relevant information based on the input query.
Augmentation: Retrieved information is added to the context provided to the language model.
Generation: The model generates a response using both its pre-trained knowledge and the retrieved information.

Advanced Techniques in Prompt Engineering – Retrieval Augmented Generation (RAG)

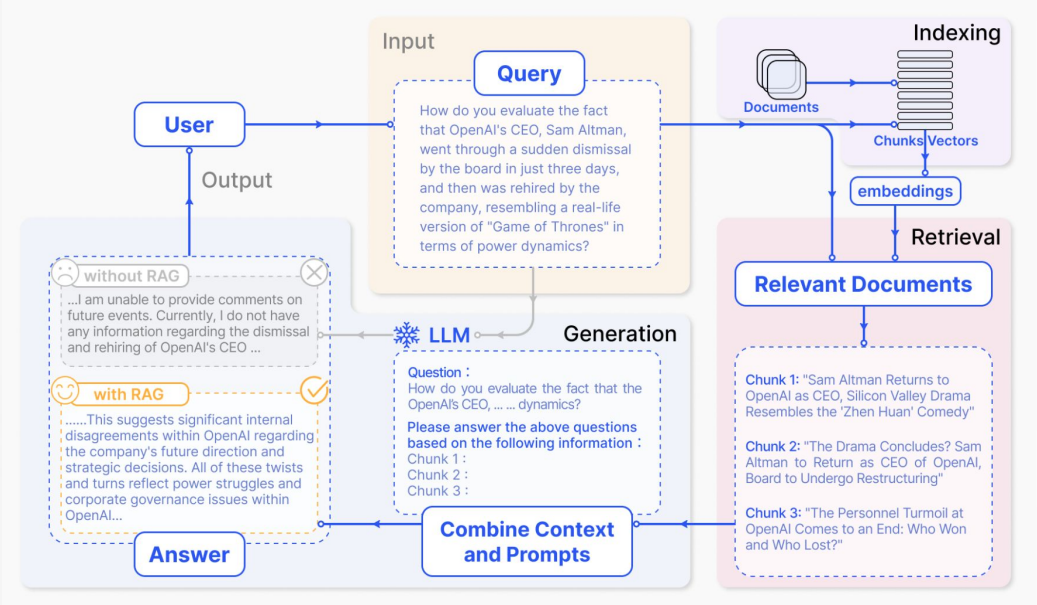
Benefits:

- Improved accuracy and up-to-date information
- Reduced hallucinations (fabricated information)
- Ability to cite sources
- Customizable knowledge base

Applications:

- Question answering systems
- Chatbots and virtual assistants
- Content generation
- Research and analysis tools

Advanced Techniques in Prompt Engineering – Retrieval Augmented Generation (RAG)



It mainly consists of 3 steps.
1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database.
2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity.
3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer

Fig 5. A representative instance of the RAG process applied to question answering. [6]

Advanced Techniques in Prompt Engineering – Autonomous Agent System

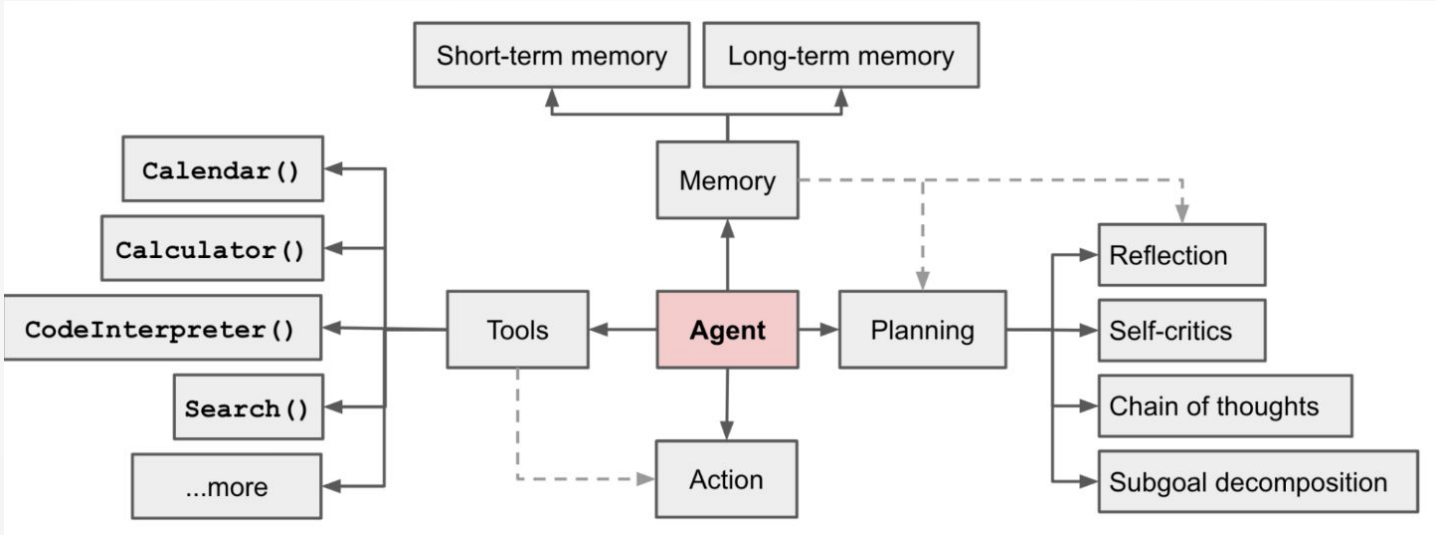


Fig 6. Overview of a LLM-powered autonomous agent system. [7]

Prompt Engineering Tools and Frameworks

Langchain:
- From Chains to Agents and web browsing
- Comprehensive suite for complex LLM applications

Semantic Kernel (Microsoft):
- Toolkit for skill development and planning
- Supports chaining, indexing, and memory access

Guidance (Microsoft):
- Modern templating language for prompt engineering
- Aligned with latest advancements

LlamaIndex:
- Specializes in data management for LLM applications
- Streamlines data integration process

Auto-GPT:
- User-friendly interface for complex LLM AI agent development

AutoGen (Microsoft):
- Capabilities in agent and multi-agent system design

PromptAppGPT, Dust, OpenPrompt, PromptFlow, PromptBase, and more…

The GenAI Life Cycle

GenAI (or Generative AI), refers to advanced machine learning systems capable of creating content, such as text, images, and even code, that is often indistinguishable from content produced by humans. The GenAI life cycle delineates the steps for creating AI-based applications, such as chatbots, virtual assistants or intelligent agents.

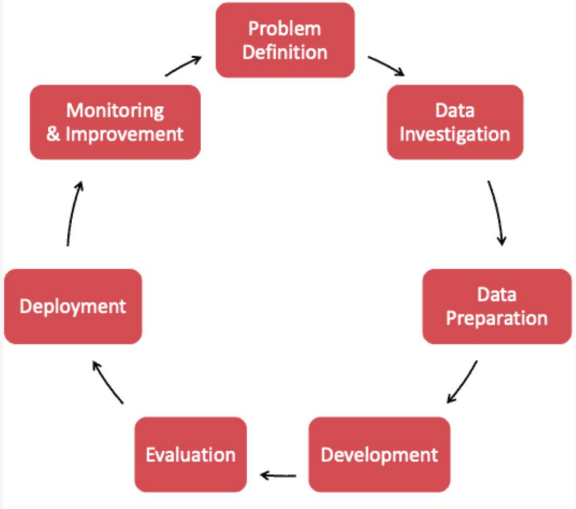


Fig 7. Illustration of The Generative AI Project Life Cycle. [8]

The GenAI Life Cycle

1. Problem Definition:
• Define problem and business context
• Set clear objectives and scope
• Determine potential impact and desired outcomes

2. Data Investigation:
• Source data for Retrieval-Augmented Generation (RAG)
• Assess data availability, relevance, and quality
• Focus on enhancing LLM capabilities with external information

3. Data Preparation:
• Clean, format, and structure data
• Process and embed data into vector store database
• Ensure compatibility with GenAI models and RAG

4. Development:
• Select and integrate appropriate LLM model(s)
• Implement RAG
• Design effective prompts
• Fine-tune LLM if necessary

5. Evaluation:
• Test for correctness, readability, performance, and reliability
• Evaluate against predefined criteria and business needs

6. Deployment:
• Set up infrastructure for hosting, scaling, and management
• Integrate with existing systems

7. Monitoring and Improvement:
• Continuously track performance, user satisfaction, and efficiency
• Update based on data, feedback, and evolving needs

References

[1]
E. Thompson, “A brief understanding of prompt engineering - Eva Thompson - Medium,” *Medium*, Sep. 14, 2023. https://medium.com/@SymeCloud/a-brief-understanding-of-prompt-engineering-b176ba9fb2ba (accessed Aug. 05, 2024).

[2]
X. Amatriain, “Prompt Design and Engineering: Introduction and Advanced Methods,” *arXiv.org*, 2024. https://arxiv.org/abs/2401.14423 (accessed Aug. 05, 2024).

[3]
C. Schmidt, “Prompt Engineering for GenAI - Beginner-Level Course: Part 1: Presentation ‘Introduction to Prompt Engineering,’” *Uvic.ca*, 2024. https://libguides.uvic.ca/Prompt_Engineering_Beginners_Course/Presentation (accessed Aug. 05, 2024).

[4]
W. X. Zhao *et al.*, “A Survey of Large Language Models,” *arXiv.org*, 2023. https://arxiv.org/abs/2303.18223 (accessed Aug. 06, 2024).

[5]
Research Graph, “The Journey of Large Language Models: Evolution, Application, and Limitations,” *Medium*, Mar. 19, 2024. https://medium.com/@researchgraph/the-journey-of-large-language-models-evolution-application-and-limitations-c72461bf3a6f (accessed Aug. 06, 2024).

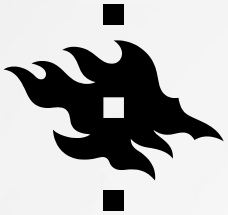
References

[6]
Y. Gao *et al.*, “Retrieval-Augmented Generation for Large Language Models: A Survey,” Mar. 2024. Available: https://arxiv.org/pdf/2312.10997

[7]
L. Weng, “LLM Powered Autonomous Agents,” *Github.io*, Jun. 23, 2023. https://lilianweng.github.io/posts/2023-06-23-agent/ (accessed Aug. 06, 2024).

[8]
J. Saltz, “The GenAI Life Cycle - Data Science Process Alliance,” *Data Science Process Alliance*, Jan. 31, 2024. https://www.datascience-pm.com/the-genai-life-cycle/ (accessed Aug. 06, 2024).

[9]
Frugal Zentennial, “Unlocking the Power of COSTAR Prompt Engineering: A Guide and Example on converting goals into system of actionable items,” *Medium*, Jan. 19, 2024. https://medium.com/@frugalzentennial/unlocking-the-power-of-costar-prompt-engineering-a-guide-and-example-on-converting-goals-into-dc5751ce9875 (accessed Aug. 06, 2024).



Thanks!