Generative AI và Prompt Engineering

Learning outcomes

#1 Hiểu về Generative AI

#2 Hiểu về prompt và cách kiến trúc prompt hiệu quả

Understand Generative AI

1, What is Generative AI?

- Generate new content (texts, images, or other media).
- Can be used with natural languages.

2, How to train Generative AI models?

2.1 Three common approaches to train ML programs

Approach	Description	Application in training GenAI models
Supervised learning	Model learns from a labeled training set. **Common applications:** - Classification - Object detection	Model learns foundational dialogue data → respond to common conversational cues appropriately.
Unsupervised learning	Model learns from an unlabeled training set and identifies data patterns and data abnormalities. **Common applications:** - Anomaly detection - Clustering/Segmentation	Model learns to interpret nuances in language, like colloquialisms.
Self-supervised learning	Model learns to predict patterns/missing parts of its input data from other parts without needing labeled examples.	Model predicts masked words or sentences within a text.

	Common applications: - NLP - Speech recognition → This training turns unsupervised learning into supervised learning by creating artificial labels from the data itself	
Reinforcement learning	Model uses feedback to make better outcomes.	Model improves responses based on users/experts' feedback.

2.2 Large Language Model:

AI model trained on large amounts of text to identify patterns between words, concepts, and phrases → generate responses to prompts (by predicting what word is most likely to come next in a sequence of words based on the word's probability).

→ **Situation**: Multiple words/chunks with the same probability → different outputs with the same prompt.

→ **Limitations factors**: Training data's quality and quantity, Prompt phrasing, Computational resources.

→ ***Higher quality, diversified data + Good prompt = Better performance of model***

3, Limitations

- Can't learn independently
- Can have biases
- Can contain inaccuracies (hallucinations) ← Knowledge cut-off (AI should recognize this limitation and answer that they do not have information after the cut-off time instead of trying to provide fault answer)

→ The need of **human's oversight** (Human-in-the-loop) & evaluation on LLM's outputs.

→ ***Suggested criteria:***

- Accurate:

Is the information provided by the model correct and factually accurate?

- Unbiased:

Does the response avoid any biased language, perspectives, or assumptions?

- Relevant:

Is the response directly related to the question asked, addressing the specific needs of the inquiry?

- Sufficient:

Does the response provide enough detail and depth to fully answer the question, without stating redundant information or lacking essential information?

4, When to apply GenAI to a task?

Answering these three questions:

- Is the task required to generate new content? (e.g: Classification + Inference = new content)
- Can the task be iterated on to achieve the best outcome? (e.g: iterated = ability to modify prompt + data)
- Are there resources to provide adequate human oversight? (e.g.: Enough experts or business users to test and give feedback to model responses?)

Prompt engineering

1, What is prompt engineering?

- Text input provides instructions to an AI model on how to generate output.
- “Prompt engineering” - design the best prompt to get the desired output.
    - An experiment: little by little
    - An iterative process
    - Have fun!

2, Good prompts = Clear & Specific

- Know what you want the LLM to produce → instruction
- Provide necessary context → context
- Structure your prompt clearly.

→ **Format of a prompt (instruction - context - request)**

****These are common parts of a prompt, but not required. Sometimes, instruction = request and context is not necessary.*

2.1 Instruction

****This is recommended based on personal experience, be flexible with your style and requirements.*

- Specific tasks or questions
- Assign a role/function/job (optional)
- Input description

2.2 Context

- Examples (shots)
- Relevant background (main purpose, timeline, restrictions,...)
- Audience
- Step guide
- Output format
- Chat history

****Select the context components based on your own challenges. NOT all listed components are required in a single prompt.*

2.3 Structure

- Use delimiter or XML tags, or markdown marks to break sections of a prompt.
- Common delimiters and their usage:

Denotations	Descriptions	Example
Brackets ([ ])	Highlight parts of the text or to specify instructions.	Provide a summary of the following text [in bullet points]: “text”

Triple Backticks (or ```)	Denote blocks of code or text, often used for longer or multi-line inputs.	Review the following code and explain its functionality:
		def calculate_area(radius):
		  import math
		  return math.pi * (radius ** 2)
		
		radius = 5
		area = calculate_area(radius)
		print(f'The area of the circle with radius {radius} is {area}')

Angle Brackets (< >)	Indicate variables or placeholders within the prompt.	Translate the following text from <source_language> to <target_language>: <Insert text here>
Asterisks *(*)**	Emphasize certain parts of the text or instructions.	Summarize this text following this guide: - Summarize part 1 - Summarize part 2 - **Do not infer**
Quotation Marks (" ")	Indicate direct quotes or specific text to focus on.	Provide a brief overview of “text from a document”.
Double Colons (::)	Separate different sections of text/content in a prompt.	Instructions:: Provide a brief overview. Content:: Text
XML Tags **<start>** **</end>**		<instruction> </instruction> <content> Text </content>
Subsection/Header in markdown: ##/###		###Instruction: Provide a brief overview of the content. ###Content: Text

3, Experiment and iterative process

3.1 Why?

Experiment with the same prompt on

- Different LLMs
- Different hyperparameter configurations
- Different words/phrases used in a prompt

The loop of output evaluation and prompt iteration.

3.2 Recommended process to build a prompt (from the middle of nowhere :):

- Start with a basic instruction.
- Add **necessary** context. (chat in a conversation until you get your ideal output)
- Summarize, clarify, or categorize all of the contexts added in the conversation into a single prompt.
- Test on different and diverse requests (from basic to advanced).
- Summarize issues
- Revise prompt (***pro tip: Use LLM to refine your prompt***)
  → **Iterative process**

3.3 Pro tips

- Use LLM to refine prompts. Template:

I'm writing a prompt for a LLM to/for ... and having a problem. Refine my prompt to solve the Problems and achieve my Expectations.
###Problem:
[Describe your problem with current prompt]
###Expectation:
[Describe your expectations with the final output]

- Be flexible with the prompt structure.
- Test on different models.
- Break down a large task/requirement into specific sub-tasks.

3.4 Exception:

- What if one prompt is unable to get the desired output?

Practice

Requirements:

- **Apply GenAI to evaluate the car damage level.**