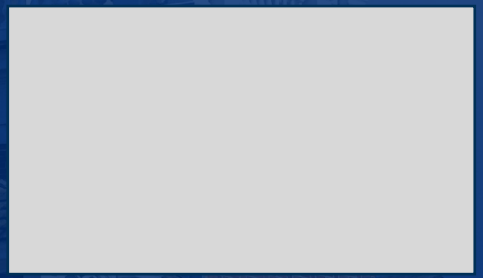
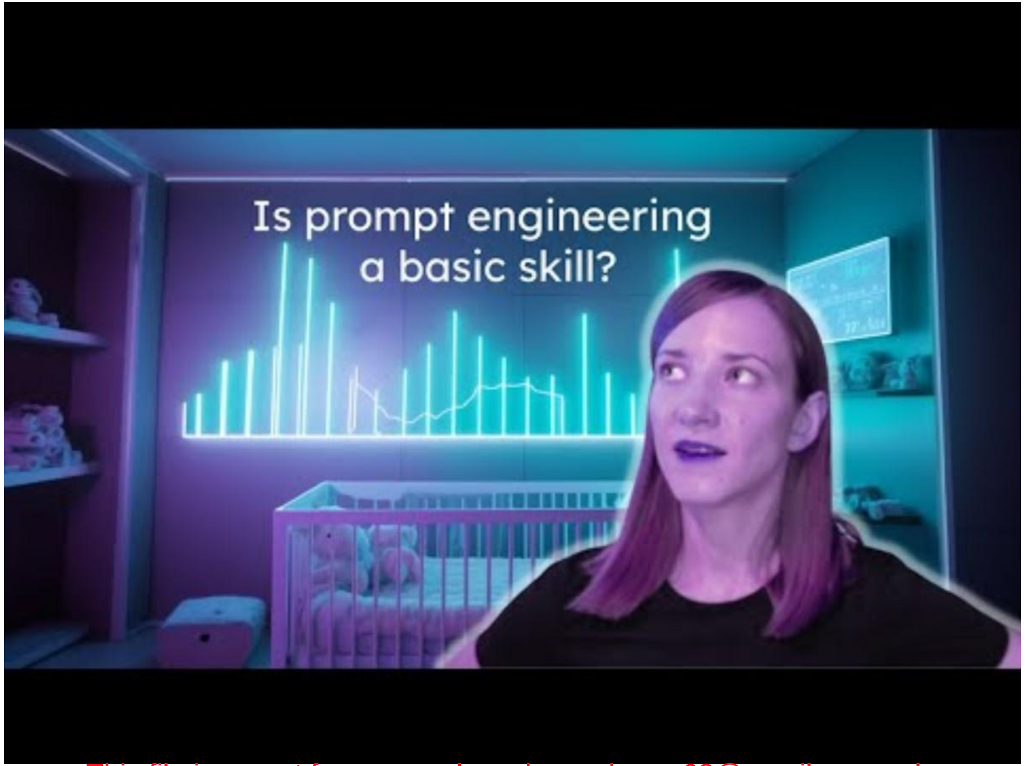
Agentic AI

Prompt Engineering: Basics



What is Prompt Engineering?

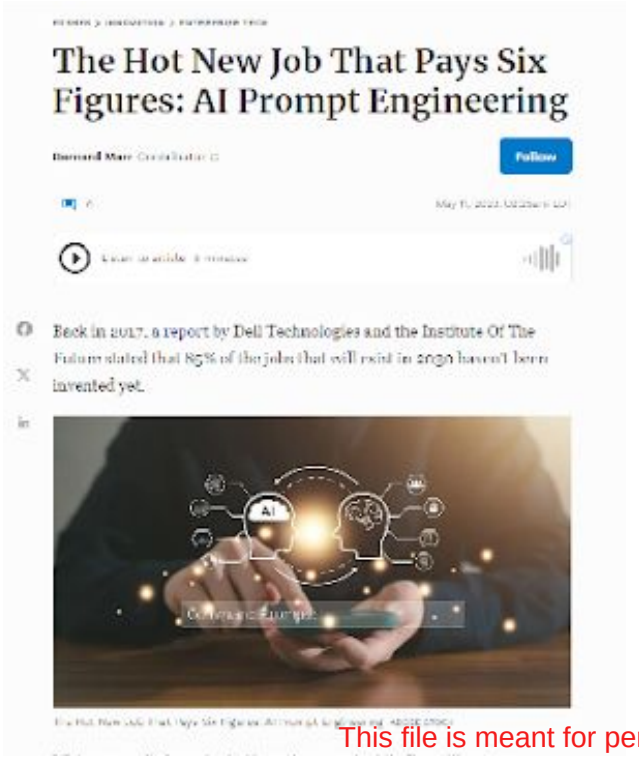


Kozyrkov, Cassie, *Is prompt engineering a basic skill? Is it even... engineering?*
https://www.youtube.com/watch?v=L2jnRk2GYwg&t=29s

What is Prompt Engineering?

- *Prompt Engineering* is the art and science of designing and structuring *prompts* (questions or tasks) fed to language models.
- It is a collection of strategies and methods to describe a task in text for an AI model.
- Improve the performance of the model on tasks.

Why Do We Engage in Prompt Engineering?



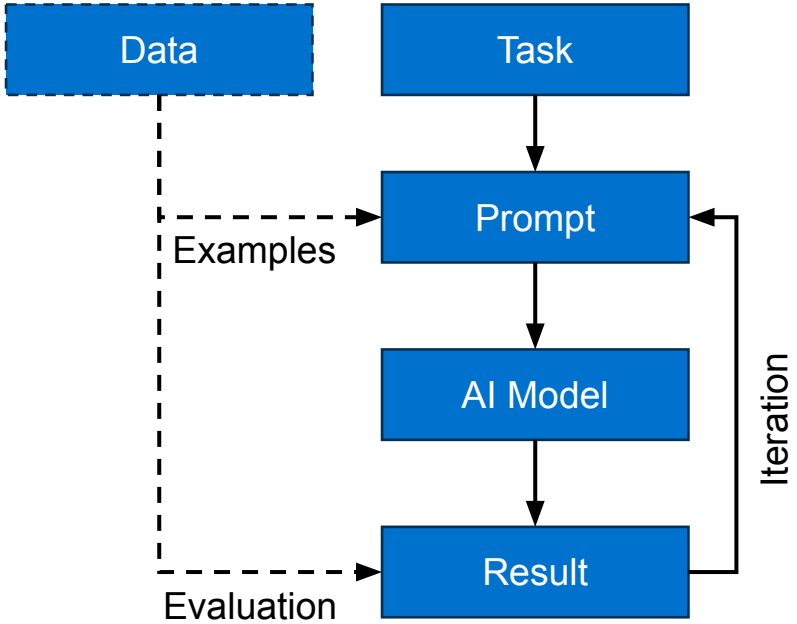
The Hot New Job That Pays Six Figures: AI Prompt Engineering

Follow

Back in 2017, a report by Dell Technologies and the Institute Of The Future stated that 85% of the jobs that will exist in 2030 haven't been invented yet.

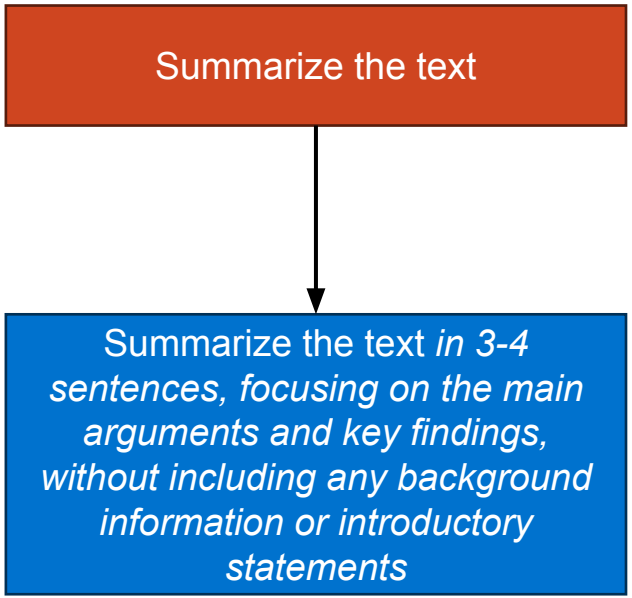
- To adapt the model to a wide range of tasks and applications, offering flexibility.
- To guide the model's responses, leading to more accurate and relevant results.
- To get the desired output efficiently, saving computational resources.

The Basic Elements of an Effective Prompt



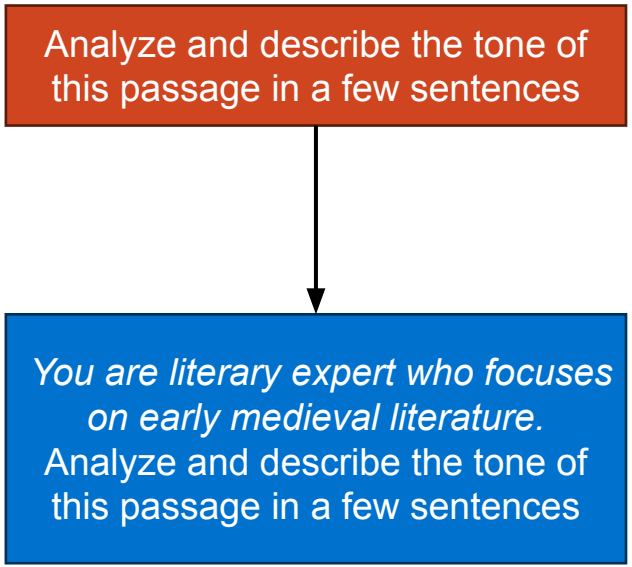
- Effective prompts normally have a couple of basic elements:
  - Clear instruction
  - Context
  - Output indicator
- Prompting frequently also benefits from additional elements:
  - Examples
  - Embodiment
  - Task breakdown

Clear Instructions in a Prompt



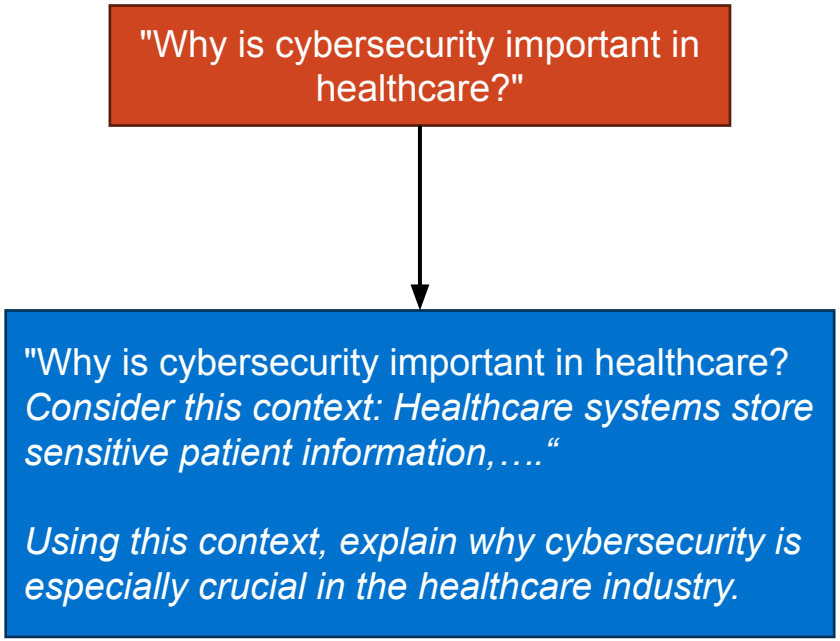
- Clearly define the task you want the model to do.
- Provide necessary details to do the task.
- Avoid vague language.
- For difficult to define tasks, providing examples (i.e., few-shot prompting) can help.

Adopting a Persona in a Prompt



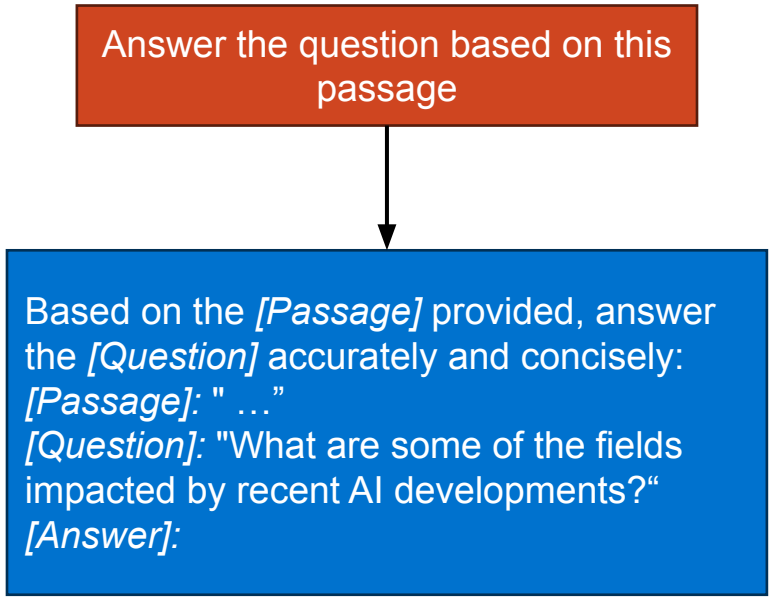
- Have the AI model play a role.
- The role should be related to the task you want the model to do.
- Providing descriptive elements to the role can also improve performance (e.g., "experienced", "from [location]", etc.).

Adding Context to a Prompt



- Provide additional details to the prompt.
  - Forms the basis of Retrieval Augmented Generation.
- Greatly helps with problems like Hallucination.
- Can be used to augment AI models with knowledge outside of their training.

Adding Text Delimiters to a Prompt



- Provide textual cues for elements of the prompt.
  - Task
  - Examples
  - Output indicator
- Cue words like "example:" or "output:" can mark particular actions fort he model.
- Delimiters like "'…"' or […] can demarcate special sections of text.

Breaking Down a Task in a Prompt

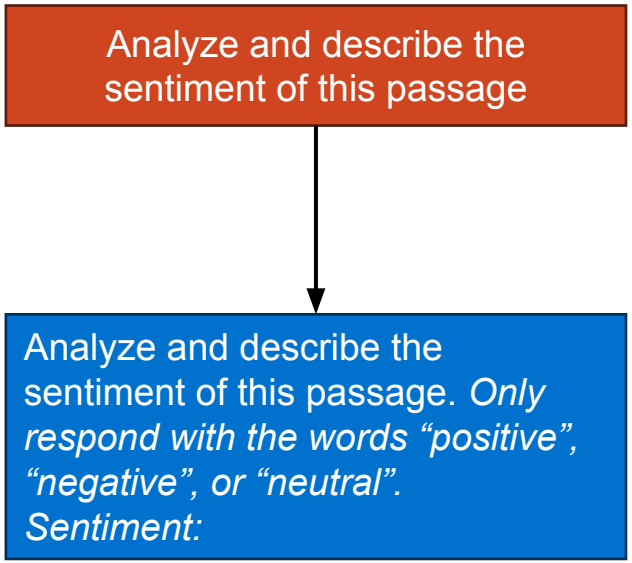
Analyze the following passage for tone, main ideas, and any notable rhetorical devices.

Analyze the following passage by completing each step below:

1. *Identify and describe the tone of the passage in one sentence.*
2. *List the two main ideas conveyed in the passage.*
3. *Highlight any rhetorical devices (e.g., metaphors, alliteration) used, and briefly explain their effect.*

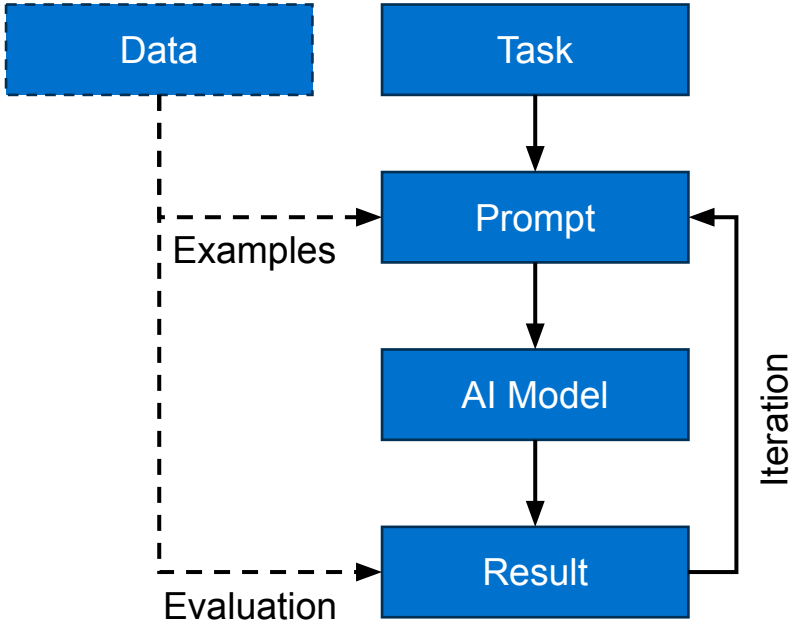
- Specify the steps required to do a task.
- Break the task into different sub-tasks which can be done over successive prompts.
- For tasks with lots of different instruction sets, classify a type of query and then use the appropriate instruction set.

Specifying Output in a Prompt



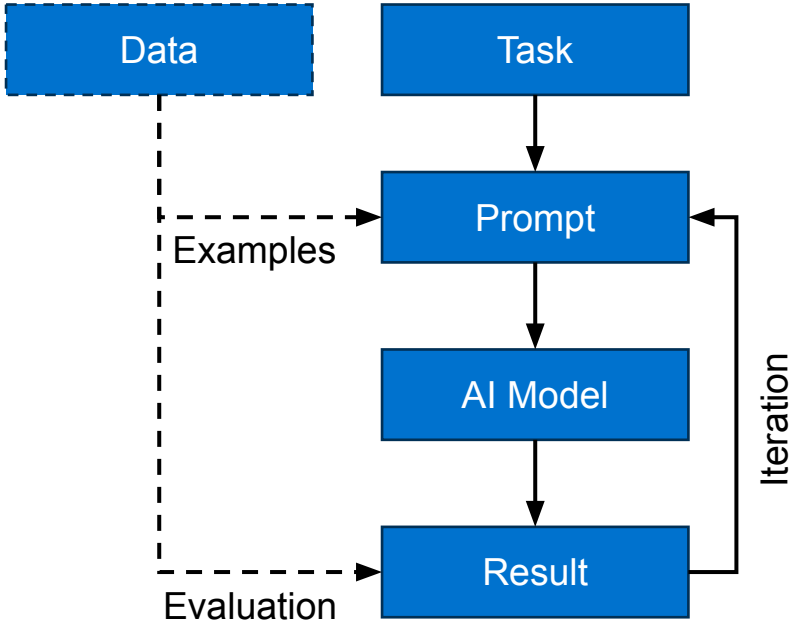
- Be descriptive about the output desired.
  - Word or sentence length
  - Special words to use (i.e., in a classification task)
- Provide an output indicator or word to cue the model (e.g., ":").

Evaluating the Quality of a Prompt

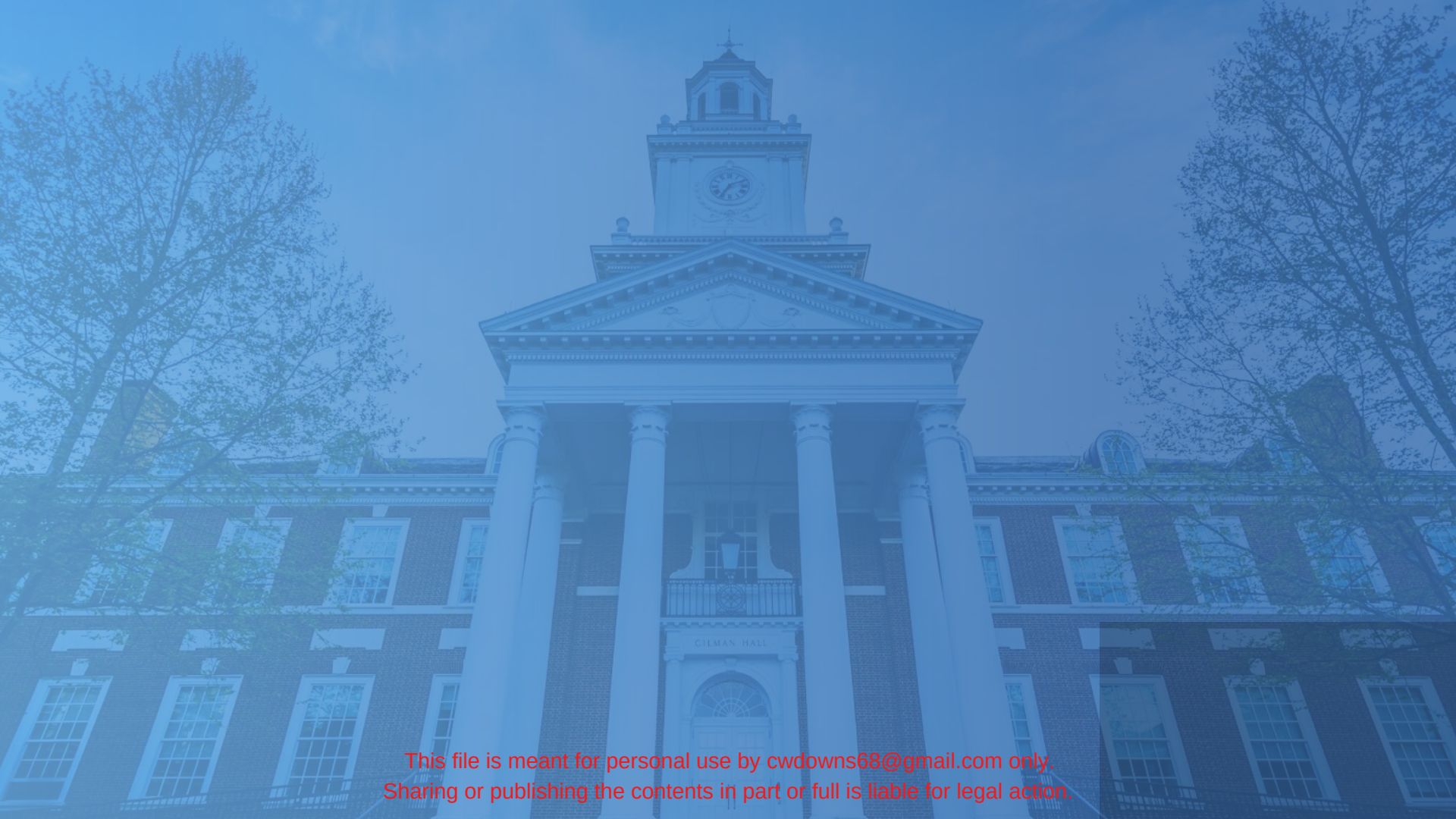


- Evaluation against performance indicators
  - Consistency
  - Completeness and specificity of the output
  - Comparison to benchmarks
- Test with variations and red-teaming
  - Ambiguous or harmful input
  - Especially for user facing applications

Bringing it All Together

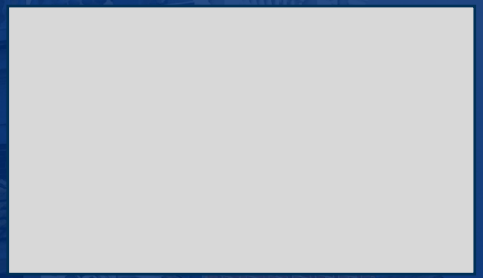


- Prompt engineering is often an iterative process.
  - It helps to have some kind of validation data of what you are looking for.
  - Test changes systematically.
- Good prompts frequently use all or many of the aspects previously described.
- Ultimately, you want to structure your inputs to AI models to optimize your outputs from them.

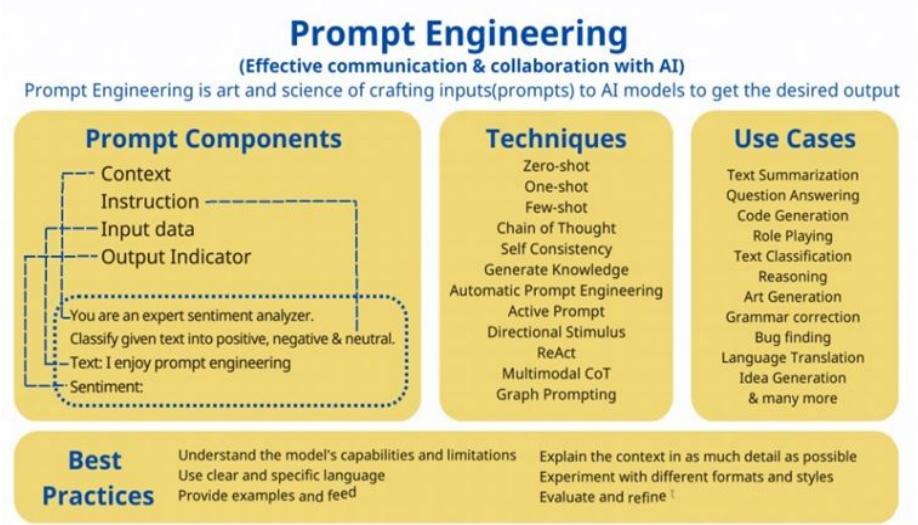


Agentic AI

Prompt Engineering: Common Design Patterns



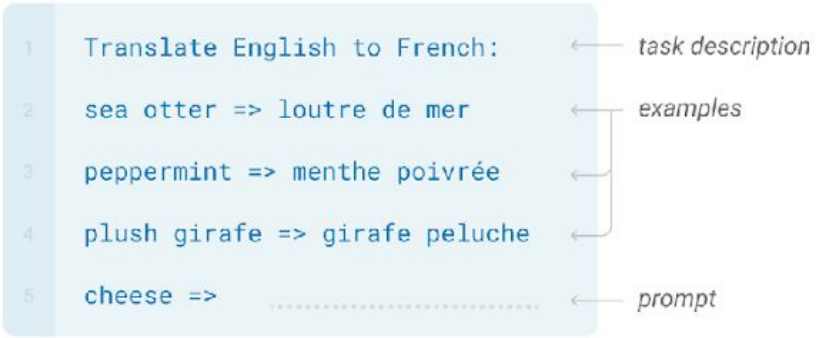
Common Prompting Design Patterns



Vaj, Tiya, *Prompt Engineer* (2024)

- There are a number of prompt formats, or design patterns, for various types of tasks.
- Some of the most successful feature:
  - Using examples and/or context data.
  - Having the AI model engage in some form of deliberate reasoning.

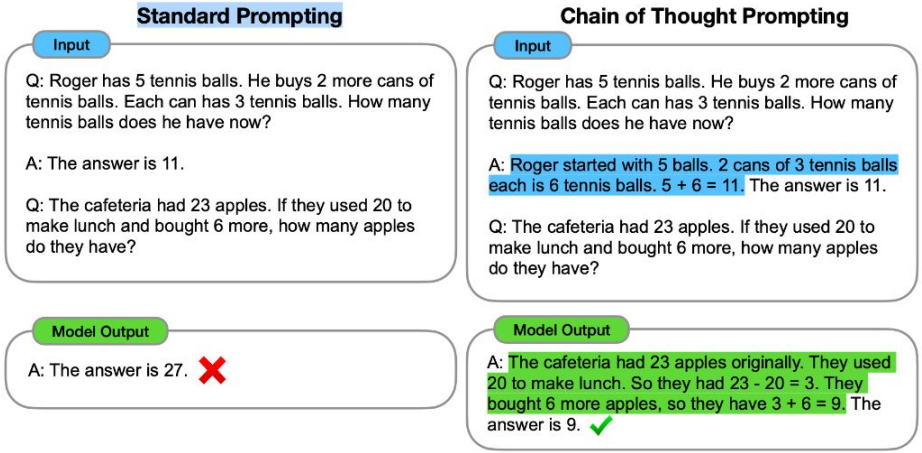
Few Shot Prompting



Brown et al., *Language Models are Few-Shot Learners* (2020)

- Provide the model with several examples of the task.
- Example: For a translation task, provide multiple pairs of sentences in two languages.
- Cautions on use:
  - The order and choice of examples can greatly influence the outcome.

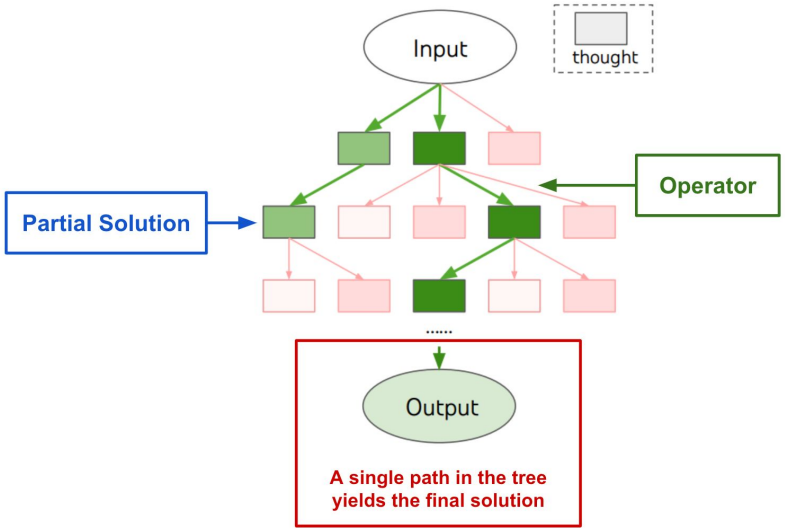
Chain-of-Thought Prompting



Wei et al., *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (2022)

- Break down complex tasks into a series of simpler tasks, guiding the model through a chain of reasoning.
- Example: For an arithmetic task, start with simple operations and build up to the more complex operation.
- Cautions on use:
  - Increased computation cost
  - Design of intermediate steps

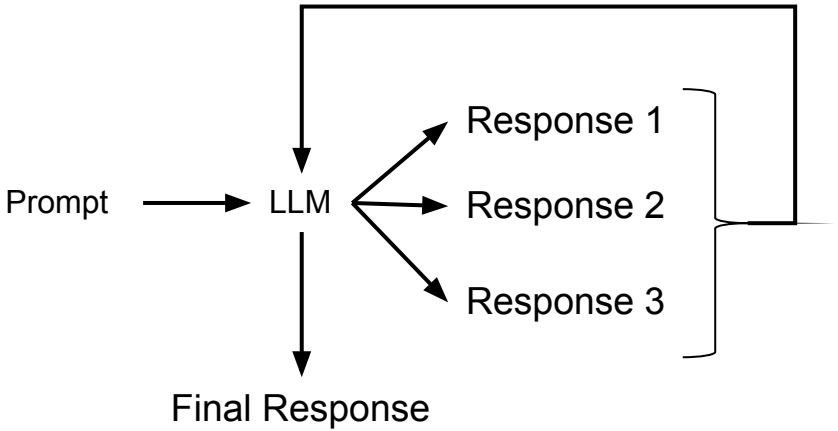
Tree-of-Thought Prompting



Yao et al., *Chain-of-Thought Prompting Elicits Reasoning in Large Language Tree of Thoughts: Deliberate Problem Solving with Large Language Models* (2023)

- Guide the model to explore multiple reasoning paths by branching into different lines of thought.
- Example: For solving a logic puzzle, prompt the model to generate multiple hypotheses and evaluate each one before converging on a solution.
- Cautions on use:
  - Increased computation cost (more than CoT)
  - Design of merge and prune steps

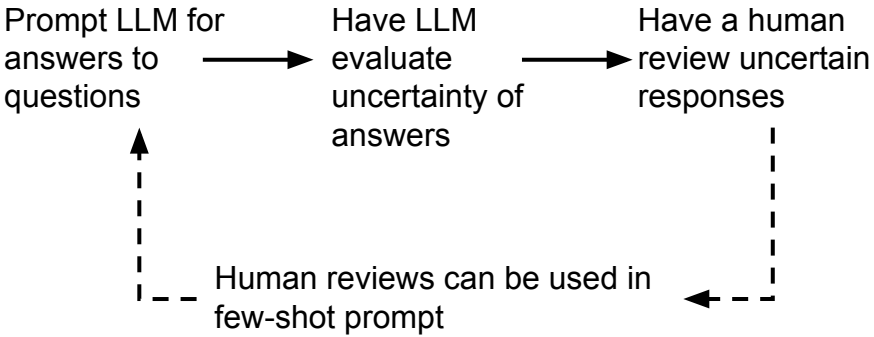
Self-Consistency Prompting



- Reminded the AI model of previous parts of the conversation or ask it to ensure its responses are consistent with prior statements.
- Example: For a question-answering task, produce several possible answers and then seek consistency across the answers.
- Cautions on use:
  - Increased computation cost
  - Risk of Majority Bias

Wang et al., *Self-consistency improves chain-of-thought reasoning in language models* (2023)

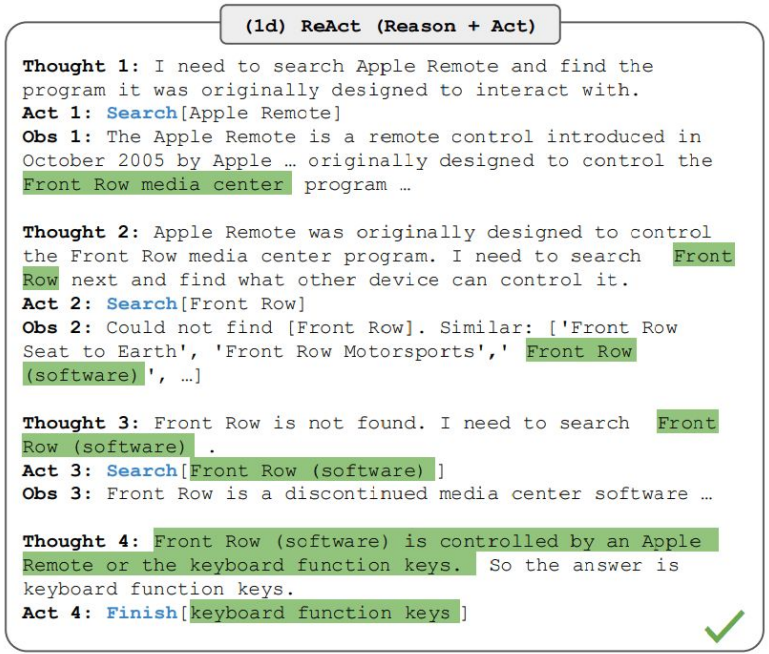
Active Prompting



Diao et al., *Active Prompting with Chain-of-Thought for Large Language Models* (2023)

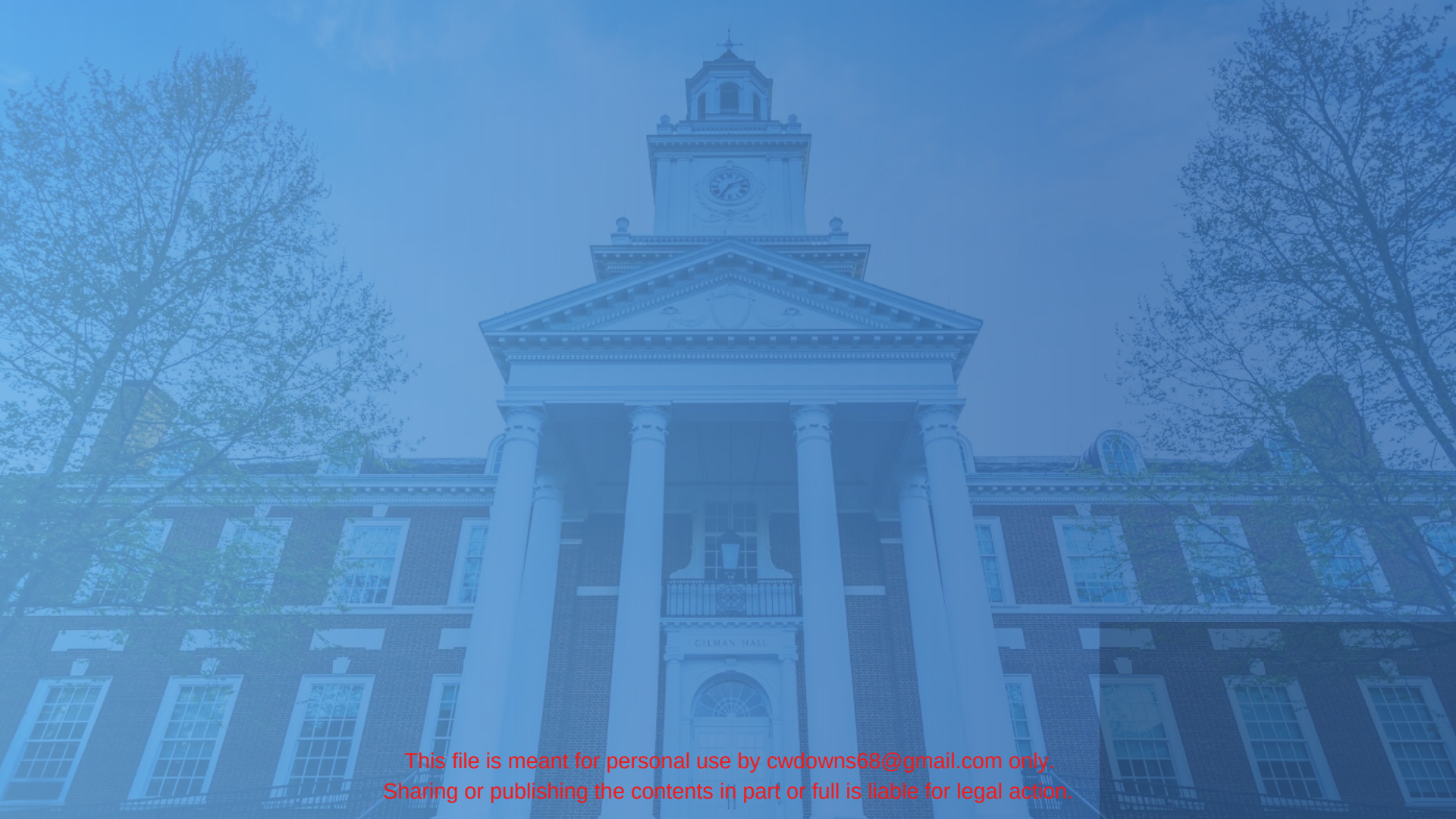
- AI model proactively seeks additional information or clarification from the user to improve its understanding of the task or prompt.
- Example: Having the model ask questions of the user about the how to classify something before doing the task
- Cautions on use:
  - Relies on human expertise

Action-Observation Prompting



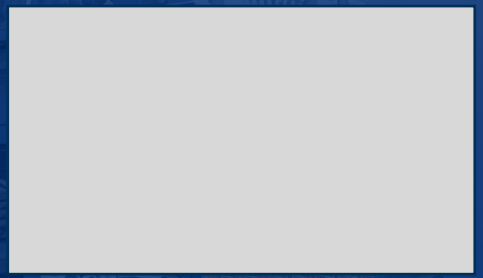
Yao et al., *ReAct: Synergizing Reasoning and Acting in Language Models* (2022)

- Alternates between performing actions (e.g., reasoning steps) and observing outcomes, using feedback to refine responses.
- Example: For a question-answering task, the model forms a hypothesis, retrieves relevant data, adjusts based on findings, and refines the answer.
- Cautions on use:
  - Complexity in prompt design

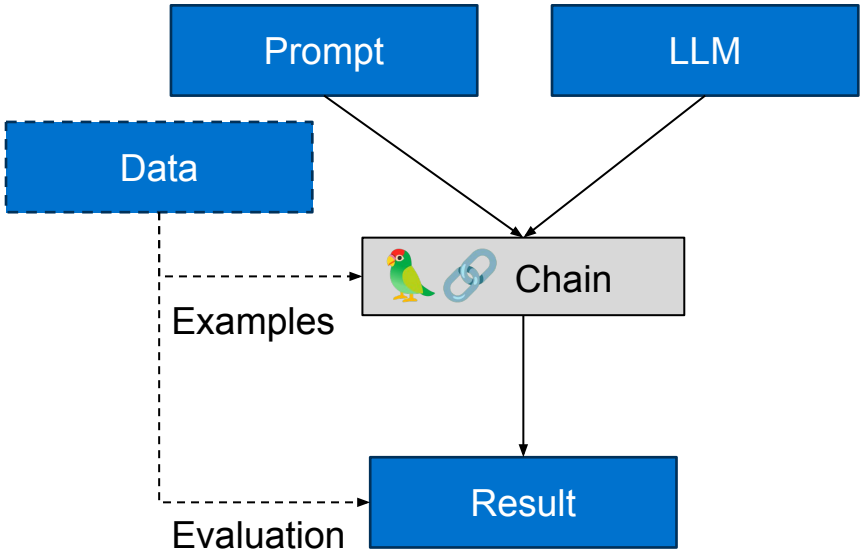


Agentic AI

Prompt Engineering: Working with LangChain 🦜🔗

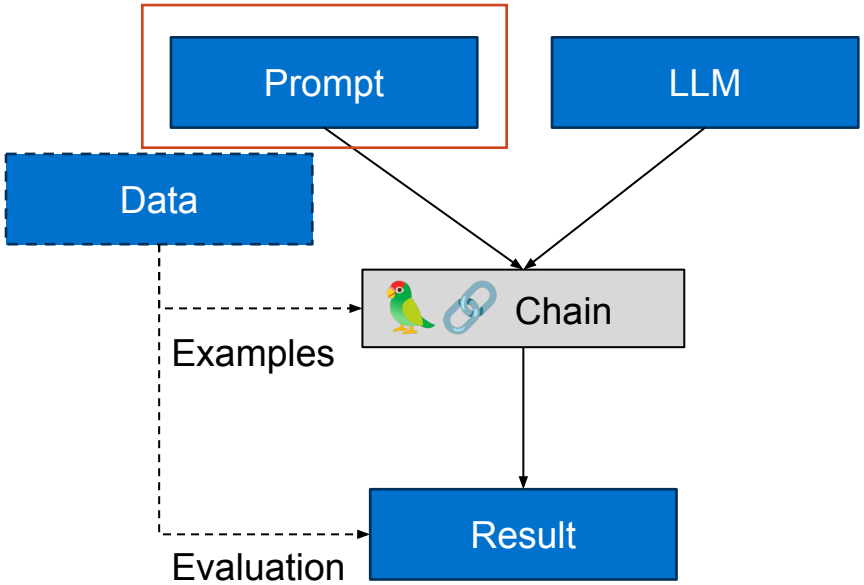


LangChain Core Concepts



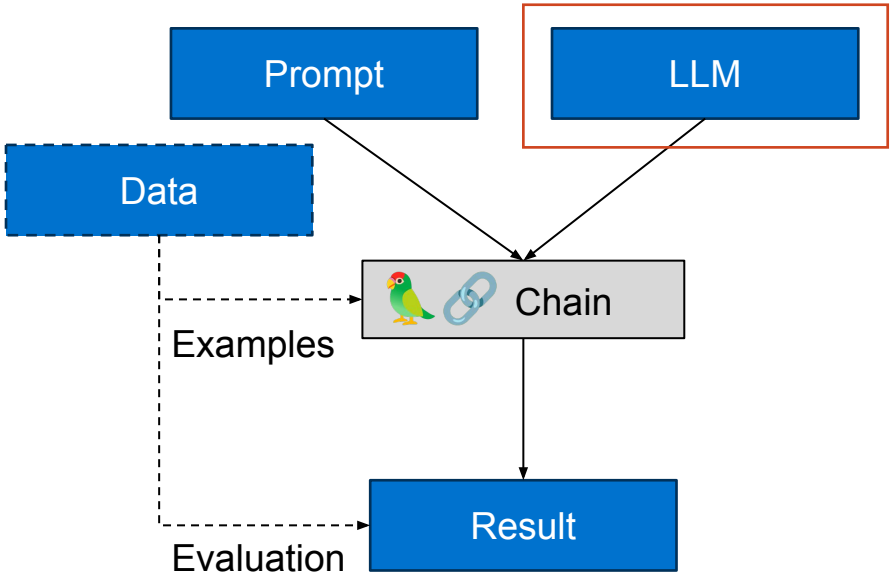
- LangChain uses the concept of "Chains" to link together things like models and prompts, and control their execution.
- For prompt engineering, the components of our chains will center around:
  - Prompt templates
  - LLMs
  - Chains

Prompt Templates



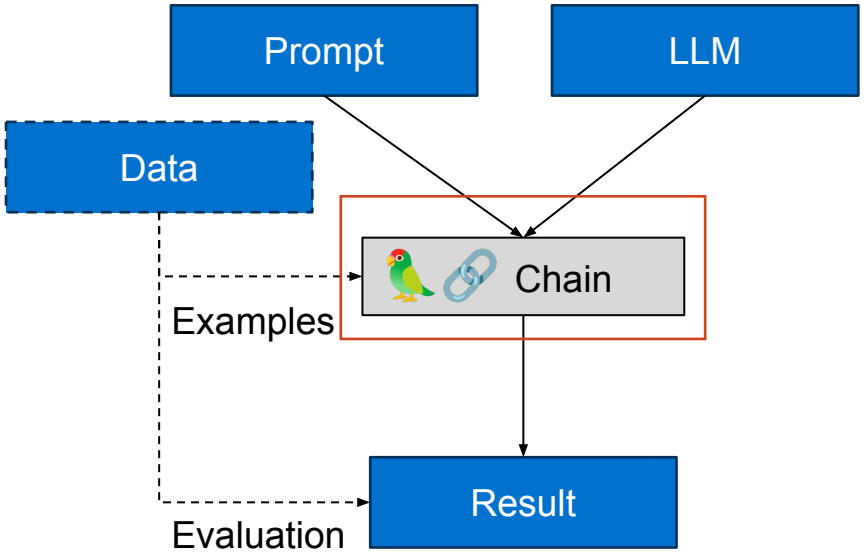
- Prompts are stored in Prompt templates, which consist of:
  - Text for the prompt with '{}' for inserting variables
  - A Prompt Template object that controls inputs and outputs
- There are a number of Prompt Template classes, but the most important are the basic BasePromptTemplate and FewShotPromptTemplate

LLMs



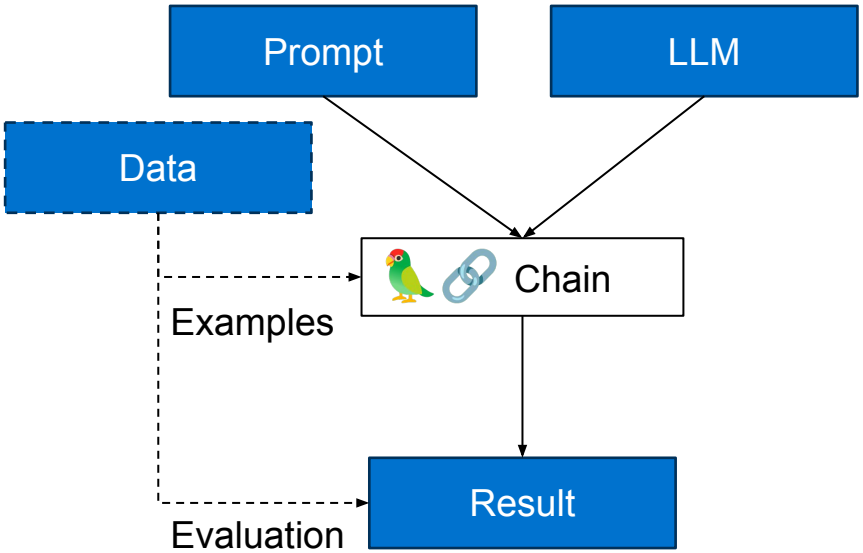
- LangChain uses an abstraction to wrap models for inclusion into Chains.
- Typically each major source of LLMs (i.e., OpenAI, Cohere, Huggingface) has a specific class for wrapping those models (or API calls to those models)
  - Model classes are also typically broken between "chat" and "llm" classes depending on the particular model

Chains

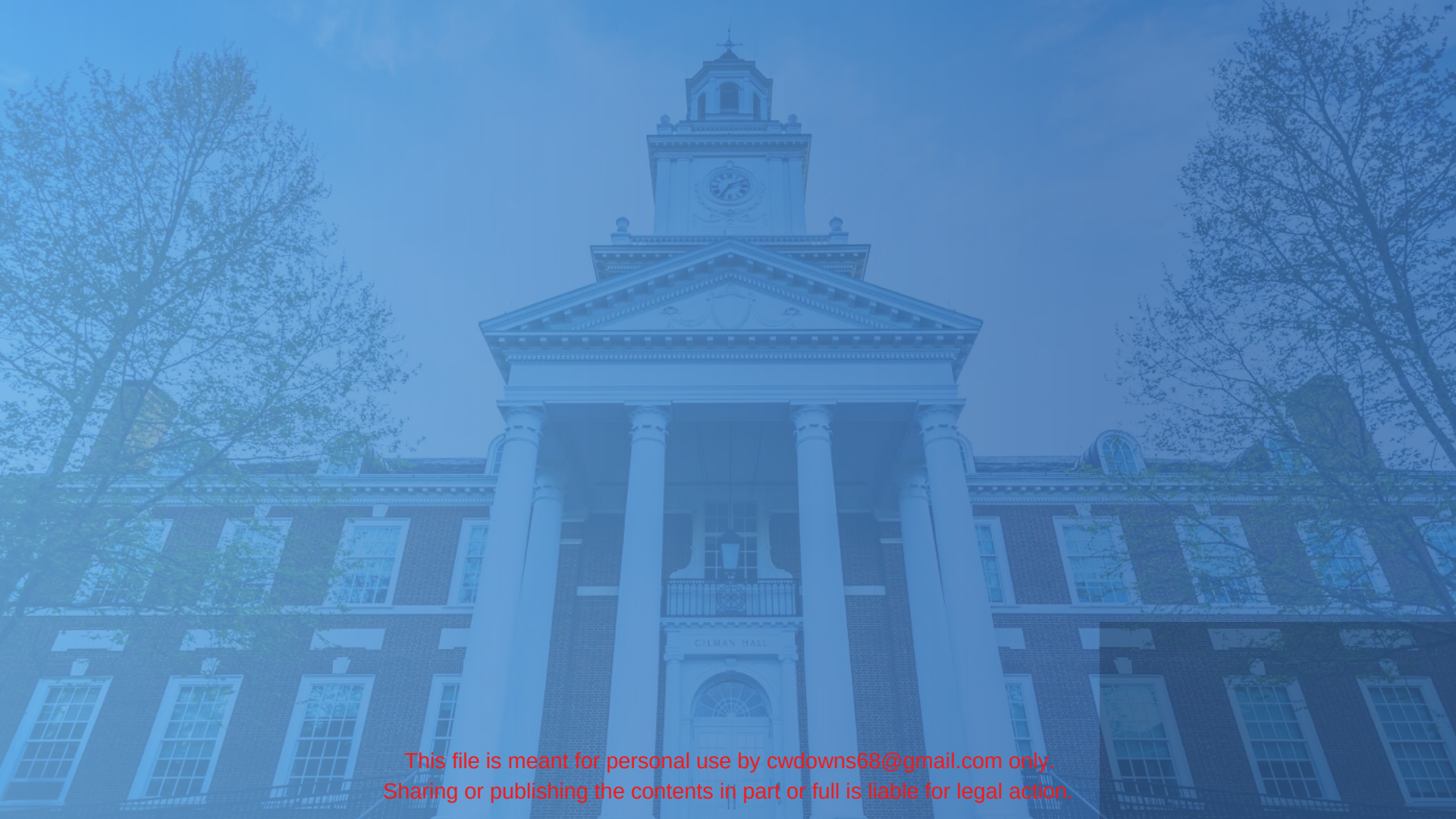


- Chains are the reusable components that link together things like LLMs and prompts
- Chains are composable with other objects or "runnables", including other chains
- Chains typically link together runnables through the "|" operator

A Final Note about LangChain



- In this section, we focused in on the elements of LangChain that most directly apply to prompt engineering
- LangChain has many other elements for things like Agents, memory, and API calls, which can also be incorporated in chains
- I invite you to explore the documentation to find other chains or components of chains!



References

- Briggs, J., & Ingham, F. (n.d.). LangChain AI Handbook. Pinecone. Retrieved from https://www.pinecone.io/learn/series/langchain/
- Kadel, S. N. (2024). Prompt Engineering: The Art of Getting What You Need From Generative AI. Ivan Allen College of Liberal Arts. Retrieved from https://iac.gatech.edu/featured-news/2024/02/AI-prompt-engineering-ChatGPT
- Cummings, L. (2023, September 26). Why AI Whispering is a Myth: There's nothing magical about structured prompt operations. Cyborgs Writing. Retrieved from https://www.isophist.com/p/the-anatomy-of-a-prompt
- OpenAI Platform Team. (n.d.). Prompt engineering. OpenAI. Retrieved from https://platform.openai.com/docs/guides/prompt-engineering
- DAIR.AI. (2024). Prompt Engineering Guide. Retrieved from https://www.promptingguide.ai/
- Diao, S., Wang, Y., Chen, Y., Zhang, R., & Tao, D. (2023). Active prompting with chain-of-thought for large language models. In Proceedings of the 37th International Conference on Machine Learning (ICML). arXiv preprint arXiv:2302.12246.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2022). ReAct: Synergizing reasoning and acting in language models. In Proceedings of the 39th International Conference on Machine Learning (ICML). arXiv preprint arXiv:2210.03629.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., ... & Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. In Proceedings of the 39th International Conference on Machine Learning (ICML). arXiv preprint arXiv:2203.11171.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. In Advances in Neural Information Processing Systems (NeurIPS), 33, 1877-1901. arXiv preprint arXiv:2005.14165.