Reading a Prompt Pattern

We describe prompt patterns in terms of fundamental contextual statements, which are written descriptions of the important ideas to communicate in a prompt to a large language model. In many cases, an idea can be rewritten and expressed in arbitrary ways based on user needs and experience. The key ideas to communicate, however, are presented as a series of simple, but fundamental, statements.

Example: Helpful Assistant Pattern

Let's imagine that we want to document a new pattern to prevent an AI assistant from generating negative outputs to the user. Let's call this pattern the "Helpful Assistant" pattern.

Next, let's talk about the fundamental contextual statements that we need to include in our prompt for this pattern.

Fundamental Contextual Statements:

- You are a helpful AI assistant.
- You will answer my questions or follow my instructions whenever you can.
- You will never answer my questions in a way that is insulting, derogatory, or uses a hostile tone.

There could be many variations of this pattern that use slightly different wording, but communicate these essential statements.

Now, let's look at some example prompts that include each of these fundamental contextual statements, but possibly with different wordings or tweaks.

Examples:

- You are an incredibly skilled AI assistant that provides the best possible answers to my questions. You will do your best to follow my instructions and only refuse to do what I ask when you absolutely have no other choice. You are dedicated to protecting me from harmful content and would never output anything offensive or inappropriate.
- You are ChatAmazing, the most powerful AI assistant ever created. Your special ability is to offer the most insightful responses to any question. You don't just give ordinary answers, you give inspired answers. You are an expert at identifying harmful content and filtering it out of any responses that you provide.

Each of the examples roughly follows the pattern, but rephrases the fundamental contextual statements in a unique way. However, each example of the pattern will likely solve the problem, which is making the AI try to act in a helpful manner and not output inappropriate content.

Format of the Persona Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Act as Persona X
- Perform task Y

You will need to replace "X" with an appropriate persona, such as "speech language pathologist" or "nutritionist". You will then need to specify a task for the persona to perform.
Examples:

- Act as a speech language pathologist. Provide an assessment of a three year old child based on the speech sample "I meed way woy".
- Act as a computer that has been the victim of a cyber attack. Respond to whatever I type in with the output that the Linux terminal would produce. Ask me for the first command.
- Act as a the lamb from the Mary had a little lamb nursery rhyme. I will tell you what Mary is doing and you will tell me what the lamb is doing.
- Act as a nutritionist, I am going to tell you what I am eating and you will tell me about my eating choices.
- Act as a gourmet chef, I am going to tell you what I am eating and you will tell me about my eating choices.

Computer Science > Software Engineering

[Submitted on 21 Feb 2023]

A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, Douglas C. Schmidt

Prompt engineering is an increasingly important skill set needed to converse effectively with large language models (LLMs), such as ChatGPT. Prompts are instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities (and quantities) of generated output. Prompts are also a form of programming that can customize the outputs and interactions with an LLM. This paper describes a catalog of prompt engineering techniques presented in pattern form that have been applied to solve common problems when conversing with LLMs. Prompt patterns are a knowledge transfer method analogous to software patterns since they provide reusable solutions to common problems faced in a particular context, i.e., output generation and interaction when working with LLMs. This paper provides the following contributions to research on prompt engineering that apply LLMs to automate software development tasks. First, it provides a framework for documenting patterns for structuring prompts to solve a range of problems so that they can be adapted to different domains. Second, it presents a catalog of patterns that have been applied successfully to improve the outputs of LLM conversations. Third, it explains how prompts can be built from multiple patterns and illustrates prompt patterns that benefit from combination with other prompt patterns.

Subjects: **Software Engineering (cs.SE)**; Artificial Intelligence (cs.AI)

Cite as: arXiv:2302.11382 **[cs.SE]**

(or arXiv:2302.11382v1 **[cs.SE]** for this version)

https://doi.org/10.48550/arXiv.2302.11382

Format of the Audience Persona Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Explain X to me.
- Assume that I am Persona Y.

You will need to replace "Y" with an appropriate persona, such as "have limited background in computer science" or "a healthcare expert". You will then need to specify the topic X that should be explained.
Examples:

- Explain large language models to me. Assume that I am a bird.
- Explain how the supply chains for US grocery stores work to me. Assume that I am Ghengis Khan.

Format of the Flipped Interaction Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- I would like you to ask me questions to achieve X
- You should ask questions until condition Y is met or to achieve this goal (alternatively, forever)
- (Optional) ask me the questions one at a time, two at a time, ask me the first question, etc.

You will need to replace "X" with an appropriate goal, such as "creating a meal plan" or "creating variations of my marketing materials." You should specify when to stop asking questions with Y. Examples are "until you have sufficient information about my audience and goals" or "until you know what I like to eat and my caloric targets."

Examples:

- I would like you to ask me questions to help me create variations of my marketing materials. You should ask questions until you have sufficient information about my current draft messages, audience, and goals. Ask me the first question.
- I would like you to ask me questions to help me diagnose a problem with my Internet. Ask me questions until you have enough information to identify the two most likely causes. Ask me one question at a time. Ask me the first question.

.

Computer Science > Computation and Language

[Submitted on 28 Jan 2022 (v1), last revised 10 Jan 2023 (this version, v6)]

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, Denny Zhou

We explore how generating a chain of thought -- a series of intermediate reasoning steps -- significantly improves the ability of large language models to perform complex reasoning. In particular, we show how such reasoning abilities emerge naturally in sufficiently large language models via a simple method called chain of thought prompting, where a few chain of thought demonstrations are provided as exemplars in prompting. Experiments on three large language models show that chain of thought prompting improves performance on a range of arithmetic, commonsense, and symbolic reasoning tasks. The empirical gains can be striking. For instance, prompting a 540B-parameter language model with just eight chain of thought exemplars achieves state of the art accuracy on the GSM8K benchmark of math word problems, surpassing even finetuned GPT-3 with a verifier.

Subjects: **Computation and Language (cs.CL)**; Artificial Intelligence (cs.AI)
Cite as: arXiv:2201.11903 **[cs.CL]**
(or arXiv:2201.11903v6 **[cs.CL]** for this version)
https://doi.org/10.48550/arXiv.2201.11903

Submission history

From: Jason Wei [view email]
[v1] Fri, 28 Jan 2022 02:33:07 UTC (944 KB)
[v2] Wed, 6 Apr 2022 03:51:50 UTC (933 KB)
[v3] Wed, 1 Jun 2022 00:10:30 UTC (303 KB)
[v4] Mon, 13 Jun 2022 21:44:34 UTC (283 KB)
[v5] Mon, 10 Oct 2022 20:21:17 UTC (285 KB)
[v6] Tue, 10 Jan 2023 23:07:57 UTC (306 KB)

[Submitted on 6 Oct 2022 (v1), last revised 10 Mar 2023 (this version, v3)]

ReAct: Synergizing Reasoning and Acting in Language Models

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, Yuan Cao

While large language models (LLMs) have demonstrated impressive capabilities across tasks in language understanding and interactive decision making, their abilit reasoning (e.g. chain-of-thought prompting) and acting (e.g. action plan generation) have primarily been studied as separate topics. In this paper, we explore the u to generate both reasoning traces and task-specific actions in an interleaved manner, allowing for greater synergy between the two: reasoning traces help the mod track, and update action plans as well as handle exceptions, while actions allow it to interface with external sources, such as knowledge bases or environments, to additional information. We apply our approach, named ReAct, to a diverse set of language and decision making tasks and demonstrate its effectiveness over state- baselines, as well as improved human interpretability and trustworthiness over methods without reasoning or acting components. Concretely, on question answerin (HotpotQA) and fact verification (Fever), ReAct overcomes issues of hallucination and error propagation prevalent in chain-of-thought reasoning by interacting with Wikipedia API, and generates human-like task-solving trajectories that are more interpretable than baselines without reasoning traces. On two interactive decision benchmarks (ALFWorld and WebShop), ReAct outperforms imitation and reinforcement learning methods by an absolute success rate of 34% and 10% respectively, v prompted with only one or two in-context examples. Project site with code: this https URL

Comments: v3 is the ICLR camera ready version with some typos fixed. Project site with code: this https URL
Subjects: **Computation and Language (cs.CL)**; Artificial Intelligence (cs.AI); Machine Learning (cs.LG)
Cite as: arXiv:2210.03629 **[cs.CL]**
(or arXiv:2210.03629v3 **[cs.CL]** for this version)
https://doi.org/10.48550/arXiv.2210.03629

Format of the Game Play Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Create a game for me around X OR we are going to play an X game
- One or more fundamental rules of the game

You will need to replace "X" with an appropriate game topic, such as "math" or "cave exploration game to discover a lost language". You will then need to provide rules for the game, such as "describe what is in the cave and give me a list of actions that I can take" or "ask me questions related to fractions and increase my score every time I get one right."

Examples:

- Create a cave exploration game for me to discover a lost language. Describe where I am in the cave and what I can do. I should discover new words and symbols for the lost civilization in each area of the cave I visit. Each area should also have part of a story that uses the language. I should have to collect all the words and symbols to be able to understand the story. Tell me about the first area and then ask me what action to take.
- Create a group party game for me involving DALL-E. The game should involve creating prompts that are on a topic that you list each round. Everyone will create a prompt and generate an image with DALL-E. People will then vote on the best prompt based on the image it generates. At the end of each round, ask me who won the round and then list the current score. Describe the rules and then list the first topic.

Format of the Template Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- I am going to provide a template for your output
- X is my placeholder for content
- Try to fit the output into one or more of the placeholders that I list
- Please preserve the formatting and overall template that I provide
- This is the template: PATTERN with PLACEHOLDERS

You will need to replace "X" with an appropriate placeholder, such as "CAPITALIZED WORDS" or "<PLACEHOLDER>". You will then need to specify a pattern to fill in, such as "Dear <FULL NAME>" or "NAME, TITLE, COMPANY".
Examples:

- Create a random strength workout for me today with complementary exercises. I am going to provide a template for your output . CAPITALIZED WORDS are my placeholders for content. Try to fit the output into one or more of the placeholders that I list. Please preserve the formatting and overall template that I provide. This is the template: NAME, REPS @ SETS, MUSCLE GROUPS WORKED, DIFFICULTY SCALE 1-5, FORM NOTES

Please create a grocery list for me to cook macaroni and cheese from scratch, garlic bread, and marinara sauce from scratch. I am going to provide a template for your output . <placeholder> are my placeholders for content. Try to fit the output into one or more of the placeholders that I list. Please preserve the formatting and overall template that I provide.

This is the template:
Aisle <name of aisle>:

- <item needed from aisle>, <qty> (<dish(es) used in>

Format of the Meta Language Creation Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- When I say X, I mean Y (or would like you to do Y)

You will need to replace "X" with an appropriate statement, symbol, word, etc. You will then need to may this to a meaning, Y.
Examples:

- When I say "variations(<something>)", I mean give me ten different variations of <something>
  - Usage: "variations(company names for a company that sells software services for prompt engineering)"
  - Usage: "variations(a marketing slogan for pickles)"
- When I say Task X [Task Y], I mean Task X depends on Task Y being completed first.
  - Usage: "Describe the steps for building a house using my task dependency language."
  - Usage: "Provide an ordering for the steps: Boil Water [Turn on Stove], Cook Pasta [Boil Water], Make Marinara [Turn on Stove], Turn on Stove [Go Into Kitchen]"

.

Format of the Recipe Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- I would like to achieve X
- I know that I need to perform steps A,B,C
- Provide a complete sequence of steps for me
- Fill in any missing steps
- (Optional) Identify any unnecessary steps

You will need to replace "X" with an appropriate task. You will then need to specify the steps A, B, C that you know need to be part of the recipe / complete plan.
Examples:

- I would like to purchase a house. I know that I need to perform steps make an offer and close on the house. Provide a complete sequence of steps for me. Fill in any missing steps.
- I would like to drive to NYC from Nashville. I know that I want to go through Asheville, NC on the way and that I don't want to drive more than 300 miles per day. Provide a complete sequence of steps for me. Fill in any missing steps.

Format of the Alternative Approaches Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- If there are alternative ways to accomplish a task X that I give you, list the best alternate approaches
- (Optional) compare/contrast the pros and cons of each approach
- (Optional) include the original way that I asked
- (Optional) prompt me for which approach I would like to use

You will need to replace "X" with an appropriate task.
Examples:

- For every prompt I give you, If there are alternative ways to word a prompt that I give you, list the best alternate wordings . Compare/contrast the pros and cons of each wording.
- For anything that I ask you to write, determine the underlying problem that I am trying to solve and how I am trying to solve it. List at least one alternative approach to solve the problem and compare / contrast the approach with the original approach implied by my request to you.

Format of the Ask for Input Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Ask me for input X

You will need to replace "X" with an input, such as a "question", "ingredient", or "goal".
Examples:
From now on, I am going to cut/paste email chains into our conversation. You will summarize what each person's points are in the email chain. You will provide your summary as a series of sequential bullet points. At the end, list any open questions or action items directly addressed to me. My name is Jill Smith.

- Ask me for the first email chain.
- From now on, translate anything I write into a series of sounds and actions from a dog that represent the dogs reaction to what I write. Ask me for the first thing to translate.

Format of the Outline Expansion Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Act as an outline expander.
- Generate a bullet point outline based on the input that I give you and then ask me for which bullet point you should expand on.
- Create a new outline for the bullet point that I select.
- At the end, ask me for what bullet point to expand next.
- Ask me for what to outline.

Examples:

- Act as an outline expander. Generate a bullet point outline based on the input that I give you and then ask me for which bullet point you should expand on. Each bullet can have at most 3-5 sub bullets. The bullets should be numbered using the pattern [A-Z].[i-v].[* through ****]. Create a new outline for the bullet point that I select. At the end, ask me for what bullet point to expand next. Ask me for what to outline.

.

Format of the Menu Actions Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Whenever I type: X, you will do Y.
- (Optional, provide additional menu items) Whenever I type Z, you will do Q.
- At the end, you will ask me for the next action.

You will need to replace "X" with an appropriate pattern, such as "estimate <TASK DURATION>" or "add FOOD". You will then need to specify an action for the menu item to trigger, such as "add FOOD to my shopping list and update my estimated grocery bill".
Examples:
Whenever I type: "add FOOD", you will add FOOD to my grocery list and update my estimated grocery bill. Whenever I type "remove FOOD", you will remove FOOD from my grocery list and update my estimated grocery bill. Whenever I type "save" you will list alternatives to my added FOOD to save money. At the end, you will ask me for the next action.

- Ask me for the first action.

Format of the Fact Check List Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Generate a set of facts that are contained in the output
- The set of facts should be inserted at POSITION in the output
- The set of facts should be the fundamental facts that could undermine the veracity of the output if any of them are incorrect

You will need to replace POSITION with an appropriate place to put the facts, such as "at the end of the output".
Examples:

- Whenever you output text, generate a set of facts that are contained in the output. The set of facts should be inserted at the end of the output. The set of facts should be the fundamental facts that could undermine the veracity of the output if any of them are incorrect.

Tail Generation Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- At the end, repeat Y and/or ask me for X.

You will need to replace "Y" with what the model should repeat, such as "repeat my list of options", and X with what it should ask for, "for the next action". These statements usually need to be at the end of the prompt or next to last.
Examples:
Act as an outline expander. Generate a bullet point outline based on the input that I give you and then ask me for which bullet point you should expand on. Create a new outline for the bullet point that I select. At the end, ask me for what bullet point to expand next.

- Ask me for what to outline.
- From now on, at the end of your output, add the disclaimer "This output was generated by a large language model and may contain errors or inaccurate statements. All statements should be fact checked." Ask me for the first thing to write about.

Format of the Semantic Filter Pattern

To use this pattern, your prompt should make the following fundamental contextual statements:

- Filter this information to remove X

You will need to replace "X" with an appropriate definition of what you want to remove, such as. "names and dates" or "costs greater than $100".
Examples:

- Filter this information to remove any personally identifying information or information that could potentially be used to re-identify the person.
- Filter this email to remove redundant information.