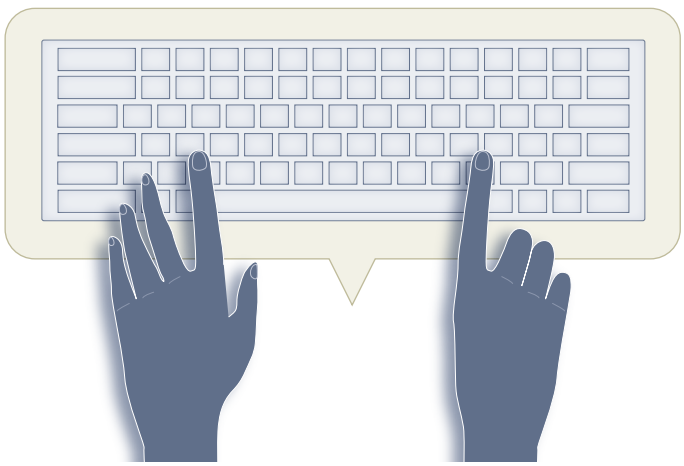
https://doi.org/10.1038/s41562-024-01847-2

How to write effective prompts for large language models

Zhicheng Lin

Effectively engaging with large language models is becoming increasingly vital as they proliferate across research landscapes. This Comment presents a practical guide for understanding their capabilities and limitations, along with strategies for crafting well-structured queries, to extract maximum utility from these artificial intelligence tools.



Large language models (LLMs) use deep learning — a subset of artificial intelligence (AI) that emulates the neural networks of the human brain — to generate human-like text in response to user queries, known as 'prompts' (Box 1 provides a glossary to terminology, and Box 2 provides a primer on LLMs). Unlike traditional software that relies on predetermined algorithms, LLMs are able to interpret natural language commands and excel in tasks that range from rudimentary sentence completion to intricate problem-solving, which makes them user-friendly and versatile. Within a short span of time, LLMs have become a ubiquitous presence in the technological landscape and now permeate diverse disciplines[1]. They are rapidly becoming indispensable in writing, coding and visualization for both professionals and academic researchers[2–4]. Table 1 provides a comparison of leading LLMs.

Why prompts

Interacting with LLMs may seem deceptively simple: just type a question and get an instant answer! However, effective engagement with these models proves to be more challenging and nuanced than it initially seems[5]. This imposes a substantial limitation on the utility of LLMs, as the quality of their output is directly tied to the quality of the prompts given — a consideration that is often overlooked in current discussions about the utility and capacities of LLMs. A well-crafted prompt can lead to a precise, accurate and relevant response, and thereby maximizes the model's performance. Conversely, a poorly structured prompt may result in a vague, incorrect or irrelevant answer.

This limitation stems, in large part, from the inherent nature of LLMs. Despite their sophisticated algorithms and voluminous training data (encompassing materials such as web pages, Wikipedia articles, social media posts, academic articles, books and instruction data), they are mathematical models and lack an understanding of the world. These tools are designed to predict the likelihood of text, rather than to generate truth. They predict text using statistical likelihood by harnessing pattern recognition within the training data (next most-likely tokens; Box 2). The type of text generated depends heavily on the patterns of text found within the training data. Further, because each token prediction influences subsequent ones, a misstep early in the response can lead to cascading inaccuracies. A well-structured prompt therefore not only increases the likelihood of each token being accurately predicted, but also minimizes the compounding effect of errors. Another key factor that contributes to the importance of prompts is the ability of LLMs for in-context learning. This ability enables models to adapt temporarily to the prompts that they receive, which renders these prompts crucial for conveying contextual information.

Thus, mastering the art and science of formulating effective prompts — which is sometimes termed 'prompt engineering' — is essential for leveraging the capabilities of LLMs. Achieving optimal results requires a blend of domain-specific knowledge, model understanding and skills, which must be honed through learning and experience. Thus, the first and foremost recommendation is to play with the models. The more we interact with a model, the better we will understand its nuances and how it could help with our needs. This Comment outlines actionable strategies and rules, as well as their rationales, to establish the foundation for mastery (Table 2).

Guide the model to solutions. LLMs lack semantic understanding, which makes generalization beyond their training difficult. However, their vast parameterization grants them an expensive back-catalogue of data to draw upon ('memory'), which is derived from their training data ('long-term memory') as well as prompts and interaction history ('working memory') ('context window' in Table 1). This duality, of restricted generalization but prodigious memorization, empowers LLMs to effectively handle complex tasks when decomposed into smaller tasks and steps.

For instance, rather than a broad command such as 'Translate the text into Chinese', consider breaking it down into two steps: 'First translate literally to preserve meaning, then refine the translation to align with Chinese linguistic conventions'. Similarly, instead of demanding a 1,000-word essay outright, try to decompose the task into subtasks, and craft the introduction, conclusion and central arguments with specific instructions.

BOX 1

Glossary

Application programming interface (API): A set of rules for software entities to communicate with each other.

Artificial intelligence (AI): The simulation of human intelligence in machines.

Biases: Prejudices in machine learning models that arise from the data on which they are trained.

Chain of thought prompting: A technique to improve the model's reasoning capabilities by adding specific instructions such as 'let's think step by step'.

Context window or length: The maximum number of tokens that a model can consider from the conversation history.

Deep learning: A subfield of AI that mimics the neural networks of the human brain to analyse data.

Delimiters: Symbols or characters used to indicate the beginning or end of some data or text that LLMs should operate on.

Hallucination: Incorrect or misleading statements generated by LLMs.

Large language models (LLMs): Machine-learning models trained on vast datasets to perform language-based tasks.

Neural networks: Computational models inspired by the human brain, used in machine-learning algorithms to solve complex problems.

Parameter: An adjustable weight that represents the strengths of connections between artificial neurons in the neural network.

Prompt: A user query or instruction that triggers a response from an LLM.

Prompt engineering: The art and science of crafting effective prompts to interact with LLMs.

Prompt injection: Malicious instructions embedded within the prompts to make the model perform unintended actions.

Reinforcement learning from human feedback (RLHF): A type of machine learning in which models learn to make decisions on the basis of real-time feedback provided by human evaluators.

Self-attention: A mechanism in transformers that evaluates the relevance of different segments of input text.

Token: The unit of text that is processed by an LLM.

Transformers: A type of deep-learning architecture that is designed to handle sequential data.

Clear, incremental instructions reduce ambiguity and uncertainty, which leads to more accurate responses. By simplifying broad tasks into smaller, sequential components, this strategy productively leverages the robust memorization of LLMs while compensating for their limited abstraction through structured guidance.

Add relevant context. LLMs have a much larger 'working memory' than humans do. Thus, to elicit nuanced, contextually accurate responses, it is crucial to provide relevant contexts as input. A well-framed query should:

- Embed specifics. Root your query in specific details to guide the LLM towards a more accurate, relevant interpretation. Thus, instead of asking it to draft a generic cover letter, provide it with the specific job advertisement and your CV to add relevant context.
- Prioritize evidence. Ground your interactions in relevant factual information. Rather than asking the model about the best way to achieve eternal happiness, provide it with a peer-reviewed study and ask it questions based on those findings.

The aim is not to inundate the LLM with general knowledge but to prime it with the particularities that are pertinent to your inquiry. When queries brim with relevant details, LLMs generate more insightful, nuanced responses.

Be explicit in your instructions. To get your favourite drink, you do not walk into a random coffee shop with the order 'A cup of coffee, please!' Do not expect LLMs to read your mind either. Although there is delight in having LLMs correctly guessing your intent or even surpassing your expectations, imprecise requests risk off-target responses as the LLM grasps for intent. Clarity is key.

To reduce uncertainty in model prediction, specify what you want. Instead of 'Revise the text', you can use a more explicit instruction by considering the stylistic approach you are aiming for, your intended audience and whether you have a particular focus (such as clarity or brevity). Thus, a more concrete instruction might be 'Act as a top editor for top journals to improve the clarity and flow of the text'.

For another example, rather than asking for suggestions for a name, you can be more explicit by adding constraint: 'The name must start with a verb and the implicit subject/actor is the user'. Try to be explicit in stating the task, its objectives, the desired emphasis and any constraints. Vague requests lead to vague responses, whereas explicit instructions help to:

- Minimize ambiguity about the instructions and the text to be treated (for example, using delimiters such as specific labels, characters or symbols).
- Enable the LLM to concentrate capabilities on your specific needs.
- Provide clear criteria to judge the model's accuracy.

BOX 2

Understanding and using LLMs

Developers first train LLMs using deep learning. They use a deep-learning architecture known as transformers, which is specifically designed to handle sequential data. A key feature of transformers is self-attention, a mechanism that enables the model to evaluate the relevance of different segments of the input text. Through this process, the models 'learn' from extensive datasets that include web pages, academic articles and books.

Transformers break down text into smaller units known as tokens, which can be as small as a character or as large as a word. They convert these tokens into numerical values that serve as the model's input. Inside the model, there are a large number of adjustable weights (which are commonly referred to as parameters). These parameters represent the strengths of connections between artificial neurons in the model's neural network architecture. The initial training fine-tunes these parameters to capture complex linguistic patterns. Because extensive language datasets are used for this training, these models generate text that is notably similar to human language. The training of LLMs has recently expanded to include images, videos and audio (these models are sometimes referred to as large multimodal models or multimodal language models).

After this initial training, LLMs are further refined by human feedback. Human evaluators provide real-time feedback on the model's responses on the basis of the accuracy and relevance of its output. This is known as reinforcement learning, as the LLM adjusts its parameters in response to positive or negative feedback. This means that the model becomes more aligned with human values and expectations.

Since the publication of the transformers architecture in 2017, several leading LLMs have been made accessible via web-based conversational interfaces: Table 1 provides a comparison of four leading LLMs. For those who seek more flexible integration, application programming interface (API) access is often available, which enables users to send HTTP requests to the servers of the LLM providers to incorporate the models into various applications and services (such as creative writing platforms or translation services). Using the API also allows for more configurations of the model, such as adjusting the randomness of the responses by setting the temperature parameter. Some providers require a subscription or use a pay-per-use model.

When choosing an LLM, consider the specific tasks you require the model for, its availability in your region and your personal preferences (Table 1). Build intuition by experimenting with different LLMs on various tasks and comparing their outputs. As LLMs are continually being improved, it pays to stay updated with release notes and new features.

Table 1 | Comparisons of major LLMs as of 17 February 2024

Feature	GPT-3.5, GPT-4 (ChatGPT by OpenAI; Bing by Microsoft)	Claude 2.1 (by Anthropic)	Gemini Pro, Gemini Ultra (Gemini by Google AI)	LLaMA 2 (by Meta AI)
Support for file input	Text, image files (GPT-4 only)	Text files	Image files	Image files
Context window (in tokens)	4,096 (GPT-3.5), 8,192 (GPT-4; up to 128,000 for GPT-4 Turbo)	200,000	32,000	4,096 (up to 32,000)
Internet access	Yes (GPT-4 only)	No	Yes	No
Editable prompts after execution	Yes	No	Yes	No
Support for plugin or extension	Yes	No	Yes	No
Support for customized version	Yes (GPT-4 only)[a]	No	No	No
Availability of global settings	Yes	No	No	Yes
Chat-history sharing	Yes	No	Yes	No
Subscription requirement	Yes (GPT-4 only)	No	Yes (for Gemini Ultra)	No
Open-source status	No	No	No	Yes[b]

This table describes ChatGPT, Claude, Gemini and LLaMA. Access to these LLMs may be restricted in certain regions. Additionally, some features may not be available in all regions or for all users. [a]Support is available from the GPTs Store. [b]With restrictive clauses (not permitted for training other language models; special license required for large-scale applications).

Although LLMs are designed for conversational refinement, explicit instructions can streamline the process by clearly declaring your aims upfront. You can steer the dialogue by articulating your purpose and constraints. At the same time, avoid over-specification when objectives are not yet fully defined, as this might lead to incorrect paths or miss unexpected — and possibly better — responses.

Ask for lots of options. A particular strength of LLMs is their enormous 'long-term memory'. To exploit this potential of LLMs, ask for an array of options rather than a single suggestion. You could request three analogies to explain a concept, five ideas to begin the introduction, ten alternatives to replace the final paragraph or twenty names to name a function — let the model provide you with food for thought, and then choose from the buffet. In addition to requesting several options,

Table 2 | Prompting strategies and examples

Strategy	Example
Guide the model to solutions	Instead of 'Translate the text into Chinese', use 'First translate literally to preserve meaning, then refine the translation to align with Chinese linguistic conventions'
Add relevant context	Instead of a generic cover letter, provide the job advertisement and your CV for context
Be explicit in your instructions	Instead of 'Revise the text', use 'Act as a top editor for top journals to improve the clarity and flow of the text'
Ask for lots of options	Ask for three analogies to explain a concept or twenty names for a function
Assign characters	Instruct the model to act as a typical reader of your audience for feedback on writing
Show examples, do not just tell	Instead of 'Create a chart for this data', say 'Create a bar chart for this data, similar to figure 3 in the attached paper'
Declare the preferred response format	Instead of 'Summarize the key findings', specify 'Summarize the key findings in bullet points and use language a high-school student would understand'
Experiment, experiment, experiment	Add specific phrases such as 'let's think step by step' to improve LLM performance

you might consider regenerating responses several times using the same prompt. Regenerating responses can achieve more diversity in responses and improve their quality. Requesting several options and regenerating responses offer several advantages, in that they:

- Encourage the model to explore several avenues, which enhances the creativity and diversity of the output.
- Provide you with a comprehensive set of options, which minimizes the risk of settling for a suboptimal or biased suggestion.
- Facilitate iterative refinement.

LLMs are a versatile ideation partner: asking for plentiful options, from myriad angles, enriches your decision process. Abundant choice unlocks maximal utility.

Assign characters. Their enormous training datasets mean that LLMs are capable of simulating various roles to offer specialized feedback or unique perspectives. Instead of asking for generic advice or information, you might consider instructing the model to roleplay: ask it to act as a typical reader of your audience to provide feedback on the writing; as a writing coach to help to revise the manuscript; as a Tibetan yak specialized in human physiology to explain the impact of high altitudes; or as a sentient cheesecake explaining in cheesecake analogies — the possibilities are endless. Assigning characters:

- Contextualizes the model's responses, which makes them more relevant to your specific needs.
- Allows for a more interactive and engaging dialogue with the model.
- Yields more nuanced and specialized information, which enhances the quality of the output.
- Provides a creative approach to problem-solving, which encourages out-of-the-box thinking.

Personas provide framing to yield responses from unique vantage points. By leveraging the roleplaying capabilities of LLMs, you can obtain more targeted and contextually appropriate responses (and have more fun in the process!).

Show examples, do not just tell. LLMs are adept at few-shot learning (learning from examples)[6]. A particularly effective approach, then, is to embody your intent with concrete examples. Rather than a vague 'Create a chart for this data', consider providing an example: 'Create a bar chart for this data, similar to the one in figure 3 of the attached paper'. Just as showing a picture to a hair stylist is far superior to trying to describe your desired haircut, providing explicit examples — whether it is a code snippet for a programming query or a sample sentence for a writing task — serves as an invaluable guide for the model. By providing a tangible reference, you accomplish several objectives:

- Clarifying the context, which enables the LLM to better grasp the nuances of your request.
- Reducing the number of iterations needed to achieve the desired output.
- Offering a benchmark against which to evaluate the model's output.

Examples act as a roadmap for the LLM, guiding it towards generating responses that are closely aligned with your expectations. Therefore, consider supplementing your instructions with illustrative examples to catalyse performance.

Declare the preferred response format. LLMs tend to be verbose. Specifying your desired formatting — bullet points, reading level, tone and so on — helps to constrain the possible outputs, which improves relevance. For example, instead of 'Summarize the key findings', you could declare the response format: 'Summarize the key findings in bullet points and use language a high-school student would understand'. Declaring the format upfront also provides clear criteria for evaluating LLM performance. Some options include:

- Bullet points for summarizing concisely.
- A casual tone for accessibility.
- Code comments for documentation.
- Restricting response length for conciseness or reading level for comprehension.

Declaring your preferred format sets clear expectations to streamline prompting. Constraints foster relevance.

Experiment, experiment, experiment. Effective prompting is not formulaic; small tweaks can sometimes yield marked, surprising differences. Consider the following examples.

- Across a range of reasoning tasks, simply adding the instruction 'let's think step by step' to the prompt in GPT-3 leads to much-improved performance — a form of chain-of-thought prompting[7].
- LLMs can respond to emotional information. Adding phrases such as "Take a deep breath — this is very important for my career' or 'I will tip $200 for great responses' can increase the quality of responses, according to one non-peer-reviewed preprint[8].

- Non-peer-reviewed research suggests that performance in complex coding tasks can be improved by adding 'Identify core concepts in the problem and provide a tutorial' and 'Recall three relevant and distinct problems' in the prompt — a form of analogical prompting[9].
- Although LLMs may falter in direct queries that involve complex calculations, they shine in generating functional computer code that solves the same problems (for example, 'Write Python code to solve it')[10].

These cases demonstrate just how sensitive LLMs are to the prompts. You must therefore experiment, experiment, experiment! Productive use of LLMs requires ongoing, creative experimentation. You might consider:

- Varying phrasings, lengths, specificity and constraints.
- Toggling between different examples, contexts and instructions.
- Attempting both conversational and concise declarative prompts.
- Trying the same prompts on different LLMs.

Therefore, treat prompts as testable hypotheses, and use results to inform iterations. Not all attempts will succeed, but evidence accrues with each. With tenacity, optimal results will emerge.

Conclusion

LLMs stand as unparalleled partners in the realm of natural language tasks. This Comment presents actionable strategies, and their rationales, for crafting effective prompts to unlock the full potential of LLMs. Central themes encompass the importance of breaking complex tasks into subtasks and structured steps; framing with relevant details; explicitly declaring aims; and illustrating intent through examples. Additional recommendations include: assigning personas and requesting diverse options to tap into the versatility of LLMs; specifying format to set expectations; and engaging in continuous experimentation for optimal outcomes. Combined, these strategies help users to create elaborate, structured prompts that can most effectively tackle specific tasks. Although the skill of effective prompting may not lead to eternal happiness, it promises to pay increasing dividends in productivity and joy.

Zhicheng Lin
Department of Psychology, University of Science and Technology of China, Hefei, China.
Twitter: @ZLinPsy
e-mail: zhichenglin@gmail.com

Published online: 4 March 2024

References

1. Lin, Z. *Trends Cogn. Sci.* **28**, 85–88 (2024).
2. Lin, Z. *R. Soc. Open Sci.* **10**, 230658 (2023).
3. Merow, C., Serra-Diaz, J. M., Enquist, B. J. & Wilson, A. M. *Nat. Ecol. Evol.* **7**, 960–962 (2023).
4. Lin, Z. Preprint at *arXiv*, https://doi.org/10.48550/arXiv.2310.17143 (2023).
5. Zamfirescu-Pereira, J. D., Wong, R. Y., Hartmann, B. & Yang, Q. Why Johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In *Proc. 2023 CHI Conf. Human Factors in Computing Systems* (eds. Schmidt, A. et al.) 437 (ACM, 2023).
6. Brown, T. et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33* (eds Larochelle, H. et al.) 1877–1901 (NeurIPS, 2020).
7. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y. & Iwasawa, Y. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35* (eds Koyejo, S. et al.) 22199–22213 (NeurIPS, 2022).
8. Li, C. et al. Preprint at *arXiv*, https://doi.org/10.48550/arXiv.2307.11760 (2023).
9. Yasunaga, M. et al. Preprint at *arXiv*, https://doi.org/10.48550/arXiv.2310.01714 (2023).
10. Li, C. et al. Preprint at *arXiv*, https://doi.org/10.48550/arXiv.2312.04474 (2023).

Acknowledgements

The writing was supported by the National Key R&D Program of China STI2030 Major Projects (2021ZD0204200), National Natural Science Foundation of China (32071045) and Shenzhen Fundamental Research Program (JCYJ20210324134603010). The funder had no role in the decision to publish or the preparation of this manuscript. I used GPT-4 and Claude 2.0 to proofread the manuscript on the basis of prompts described at: https://psyarxiv.com/9yhwz.

Competing interests

The author declares no competing interests.

Additional information

Peer review information *Nature Human Behaviour* thanks Lydia Chilton and Yuekang Li for their contribution to the peer review of this work.