GPT Guide

Aurimas Aleksandras Nausėdas

Agenda

1. ChatGPT
2. GPT4
3. Comparison
4. Principles
5. Tips
6. 10 GPT prompts for Programming
7. Note
8. Conclusion

Terms

- ChatGPT
- GPT4
- NLP
- LLM
- Prompt

What is ChatGPT?

ChatGPT is based on the GPT architecture, which stands for Generative Pre-trained Transformer. As a language model, it is designed to understand and generate human-like text based on the input it receives.

What is GPT4?

GPT-4 is the fourth iteration of OpenAI's GPT series and a successor to ChatGPT. It is an advanced and more powerful version of the GPT architecture, building upon the strengths of its predecessors.

Comparison

Some key differences between ChatGPT and GPT-4 are:

- Parameters: GPT-4 has more parameters than ChatGPT.
- Focus: ChatGPT is a conversational AI while GPT4 more general-purpose language
- Implementation: ChatGPT is an implementation of the GPT-4 architecture,
- Performance: GPT-4 outperforms ChatGPT in a wide range of tasks.
- Use cases: GPT-4 handles more complex and challenging tasks.

Principles

- **Clarity:** Use clear and specific language that is easy for the ChatGPT to understand.

- **Conciseness:** Be as concise as possible in your prompts, avoiding unnecessary words.

- **Relevance:** Make sure that your prompts are relevant to the conversation and the needs of the user.

Principles :Clear & Concise prompts

Some specific techniques for writing effective ChatGPT prompts:

1. Define the **purpose** and **focus** of the conversation.
2. Use **specific** and **relevant language**.
3. **Avoid open-ended or overly broad prompts**.
4. **Keep the conversation on track**.

By following these techniques, you can craft clear and concise ChatGPT prompts.

Tips: The „Act as ...“ Hack

One of the **most useful techniques for crafting effective ChatGPT prompts** is the **"act as"** hack. This technique involves using the phrase "act as" in the prompt to tell the ChatGPT to assume a specific role or persona in the conversation. This can be especially useful for creating more engaging and immersive conversations, or for simulating real-world scenarios.

Tips: Mistakes to avoid

Here are a few common mistakes to avoid when crafting ChatGPT prompts:

- **Overloading** the prompt with too much information
- **Using jargon** or **ambiguous** language
- Being **too vague or open-ended**
- **Neglecting to include necessary instructions or constraints**

Tips: Maintaining clarity

1. Start with a clear goal or purpose for the conversation.
2. Use specific, targeted questions instead of open-ended ones.
3. **Avoid including too much information** in a single prompt.
4. Use clear, concise language that is easy for the ChatGPT to understand.
5. Use transitional phrases to smoothly move from one topic to another.
6. **Be aware of the ChatGPT's capabilities and limitations.**
7. Test and debug your prompts to ensure that they are clear and effective.
8. **Use the "act as" hack to help the ChatGPT understand its role in the conversation.**

Tips: Asking

NEVER ask an ChatGPT/GPT4 for information you can't validate yourself, or to do a task that you can't verify has been completed correctly.

The one exception is when it's not a crucial task. For instance, asking ChatgGTP/GPT4 for apartment decorating ideas is fine.

Tips: Practises examples

Bad: "Using literature review best practices, summarize the research on breast cancer over the last ten years."

- This is a bad request because you can't directly check if it's summarized the literature correctly.

Better: "Give me a list of the top review articles on breast cancer research from the last 10 years."

- This is better because you can verify that the sources exist and vet them yourself and of course, they are authored by human experts.

Tips: Coding

It's pretty easy to ask an ChatGPT/GTP4 to write code or find relevant information for you but the quality of the responses can vary widely. Luckily, there are things you can do to improve the quality.

SET THE CONTEXT:

- Tell the LLM explicitly what information it should be using
- Use terminology and notation that biases the LLM towards the right context

Tips: Approach Example

If you have thoughts about how to approach a request tell the ChatGPT/GPT4 to use that approach.

Bad: "Solve this inequality.”

Better: "Solve this inequality using Cauchy-Schwarz theorem followed by an application of completing the square."

Tips: Sppecificity

BE SPECIFIC: This isn't google. You don't have to worry about whether there's a website that discusses your exact problem.

Bad: "How do I solve a simultaneous equation involving quadratic terms?"

Better: "Solve x=(1/2)(a+b) and y= (1/3) (a^2+ab+b^2) for a and b"

Tips: Output

DEFINE YOUR OUTPUT FORMAT. Take advantage of the flexibility of ChatGPT/GPT4 to format the output in the way that's best for you such as:

- Code
- Mathematical formulas
- An essay
- A tutorial
- Bullet points

You can even ask for code that generates:

- Tables
- Plots
- Diagrams

Tips: Validation

Once you have the output, that is only the beginning. YOU NEED TO VALIDATE THE RESPONSE. This includes:

- Finding inconsistencies
- Googling terminology in the response to get supporting sources
- Where possible, generating code to test the claims yourself

Tips: Checking

ChatGPT or GPT4 can make weird mistakes that are inconsistent with their seeming level of expertise.

For instance, ChatGPT might mention an extremely advanced mathematical concept yet fumble over simple algebra.

This is why you need to CHECK EVERYTHING.

Tips: Errors

USE THE ERRORS TO GENERATE FEEDBACK:

- If you see an error or inconsistency in the response, ask the GPT4 to explain it

- If the GPT generates code with bugs, cut and paste the error messages into the GPT4 window and ask for a fix

Tips: Ask more

ASK MORE THAN ONCE: GPTs are random. Sometimes simply starting a new window and asking your question again can give you a better answer.

Tips: References

REFERENCES References are an especially weak point for GPTs. Sometimes, the references an ChatGPT/GPT4 gives you exist and sometimes they don’t.

The fake references aren't completely useless. In my experience, the words in the fake references are usually related to real terms and researchers in the relevant field. So googling these terms can often get you closer to the information you're looking for.

Tips: Productivity

PRODUCTIVITY There are a lot of unrealistic claims that LLMs can make you 10x or even 100x more productive. In my experience, that kind of speedup only makes sense in cases where none of the work is being double-checked which would be irresponsible for me as an academic.

10 GPT prompts for programming

1. Generate boilerplate code

Prompt formula:

Generate boilerplate code for an app that **[explain what you need this app to do].** Please use **[explain what languages and frameworks should be used].**

Prompt example:

Generate boilerplate code for an app that integrates to an external API an. Please use javascript code on the express.js framework.

10 GPT prompts for programming

2. Compare frameworks/algorithms

Prompt formula:

I'm building a **new [explain what you're building]**, and want to compare **[first comparis on item]** with **[second comparison item].** Please propose the scope for a simple **[what you'rebuilding],** and generate two code bases that fulfill that scope, one using **[first comparison item]** and another using **[second comparison item]**. Please redact clear instructions for me to run both apps on my local machine.

Prompt example:

I'm building a new frontend app, and want to compare React.js with Vue.js. Please propose the scope for a simple frontend app, and generate two code bases that fulfill that scope, one using React.js and another using Vue.js. Please redact clear instructions for me to run both apps on my local machine.

10 GPT prompts for programming

3. Explain code

Prompt formula:
Explain this code to me:
[code you want to be explained]

Prompt example:
Explain this code to me:

```
fetch("https://api.stripe.com/v1/payments?created[gte]=last_month")
.then(response => response.json()).then(data => {
let transactions = data.transactions;
let groupedTransactions = {};
transactions.forEach(transaction => {
if (!groupedTransactions[transaction.entity]) {
groupedTransactions[transaction.entity] = transaction.amount;
} else { groupedTransactions[transaction.entity] += transaction.amount;}});
let sortedTransactions = Object.entries(groupedTransactions).sort((a, b) => b[1] - a[1]);
sortedTransactions.forEach(transaction => {
console.log(${transaction[0]}: $${Math.ceil(transaction[1])});});});}
```

10 GPT prompts for programming

4. Comment code

Prompt formula:
Regenerate the code snippet below, but please include comments to each line of code:
[code to be commented]

Prompt example:
Regenerate the code snippet below, but please include comments to each line of code:

```
fetch("https://api.stripe.com/v1/payments?created[gte]=last_month")
.then(response => response.json()).then(data => {let transactions = data.transactions;
let groupedTransactions = {}; transactions.forEach(transaction => {if (!groupedTransactions[transaction.entity]) {
groupedTransactions[transaction.entity] = transaction.amount;
} else { groupedTransactions[transaction.entity] += transaction.amount;}});
let sortedTransactions = Object.entries(groupedTransactions).sort((a, b) => b[1] - a[1]);
sortedTransactions.forEach(transaction => {console.log(${transaction[0]}: $${Math.ceil(transaction[1])});});});}
```

10 GPT prompts for programming

5. Generate test cases

Prompt formula:
Write test cases for **[cases to be tested]** to the below code snippet. First outline the test cases you'll write. Second, write the test cases in **[language and framework to be used to write the tests]**.
[code to be tested]

Prompt example:
Write test cases for the main edge cases that could happen to the below code snippet. First outline the test cases you'll write. Second, write the test cases in javascript using the Jest framework.

```
fetch("https://api.stripe.com/v1/payments?created[gte]=last_month")
.then(response => response.json()).then(data => {let transactions = data.transactions;
let groupedTransactions = {}; transactions.forEach(transaction => {
if (!groupedTransactions[transaction.entity]) {groupedTransactions[transaction.entity] = transaction.amount;} else {
groupedTransactions[transaction.entity] += transaction.amount;}}); let sortedTransactions = Object.entries(groupedTransactions).sort((a, b) =>
b[1] - a[1]);sortedTransactions.forEach(transaction => {console.log(${transaction[0]}: $${Math.ceil(transaction[1])});});});}
```

10 GPT prompts for programming

6. Generate documentation

Prompt formula:
Generate documentation for the code below. You should include detailed instructions to allow a developer to run it on a local machine, explain what the code does, and list vulnerabilities that exist in this code.
[code to be documented]

Prompt example:
Generate documentation for the code below. You should include detailed instructions to allow a developer to run it on a local machine, explain what the code does, and list vulnerabilities that exist in this code.

```
fetch("https://api.stripe.com/v1/payments?created[gte]=last_month").then(response => response.json()).then(data => {let transactions = data.transactions; let groupedTransactions = {}; transactions.forEach(transaction => {if (!groupedTransactions[transaction.entity]) {groupedTransactions[transaction.entity] = transaction.amount;} else { groupedTransactions[transaction.entity] += transaction.amount;}}); let sortedTransactions = Object.entries(groupedTransactions).sort((a, b) => b[1] - a[1]); sortedTransactions.forEach(transaction => {console.log(${transaction[0]}: $${Math.ceil(transaction[1])});});});
```

10 GPT prompts for programming

7. Generate regexes

Prompt formula:
Generate a regex to match **[the pattern you want to match]**

Prompt example:
Generate a regex to match an email address

10 GPT prompts for programming

8. Rewrite code using correct style

Prompt formula:
Rewrite the code below following the **[guidelines to be followed]**.
[code you want to rewrite]

Prompt example:
Rewrite the code below following the Google style guidelines for javascript.

```
fetch("https://api.stripe.com/v1/payments?created[gte]=last_month")
.then(response => response.json()).then(data => {let transactions = data.transactions;
let groupedTransactions = {}; transactions.forEach(transaction => {
if (!groupedTransactions[transaction.entity]) {groupedTransactions[transaction.entity] =
transaction.amount;} else { groupedTransactions[transaction.entity] += transaction.amount;}});
let sortedTransactions = Object.entries(groupedTransactions).sort((a, b) => b[1] - a[1]);
sortedTransactions.forEach(transaction => {console.log(${transaction[0]}:
$${Math.ceil(transaction[1])});});});}
```

10 GPT prompts for programming

9. Find bugs in code

Prompt formula:
Please find the bug in the code below. This is what it should be doing: **[outline of the desired functionality]** Code: **[code to be debugged]**

Prompt example:
Please find the bug in the code below. This is what it should be doing:
1. Fetch the response from the stripe API for payments received last month. 2. Parse the response json into an arrays with all transactions. 3. Traverse the array to group all transactions from the same entity, and sums their amounts. The result is stored in a different array. 4. Sort the resulting array by amount received, descending. 5. Write to the console all payments, sorted by date, with money amounts rounded up to the integer.
Code: fetch("https://api.stripe.com/v1/payments?created[gte]=last_month").then(response => response.json()).then(data => {let transactions = data.transactions; let groupedTransactions = {}; transactions.forEach(transaction => {if (!groupedTransactions[transaction.entity]) {groupedTransactions[transaction.entity] = transaction.amount;} else { groupedTransactions[transaction.entity] += transaction.amount;}}); let sortedTransactions = Object.entries(groupedTransactions).sort((a, b) => b[1] - a[1]); sortedTransactions.forEach(transaction => {console.log(${transaction[0]}: $${Math.ceil(transaction[1])});});});}

10 GPT prompts for programming

10. Solve leetcode type algorithms

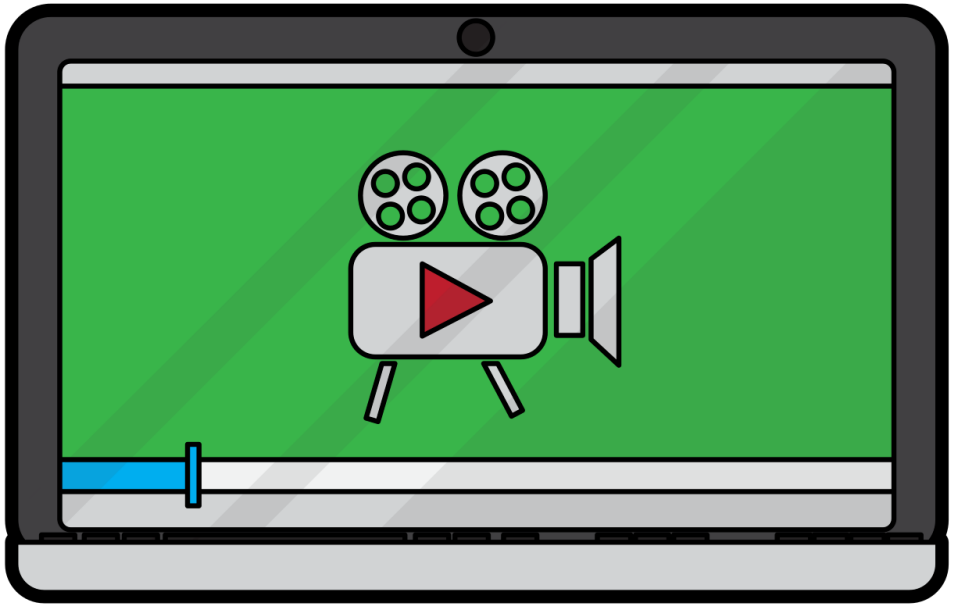
Prompt formula:
Generate code in **[desired language]** to solve the following challenge:
[outline of the challenge to be solved]

Prompt example:
Generate code in javascript to solve the following challenge:
● We have one 2D array, filled with zeros and ones. We have to find the starting point and ending point of all rectangles filled with 0. It is given that rectangles are separated and do not touch each other however they can touch the boundary of the array. A rectangle might contain only one element.
Example array: Input = [[1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 0, 0, 0, 1], [1, 1, 1, 0, 0, 0, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1]]

Conclusion

Here are a few suggestions for how you can continue to improve your skills with ChatGPT & GPT4:

1. **Practice, practice, practice!**
2. Seek feedback from others.
3. Learn from others
4. Experiment with different styles and approaches.
5. Stay up-to-date on the latest developments in ChatGPT and artificial intelligence



Videos:

https://www.youtube.com/watch?v=NcCNw_UXnOc
https://www.youtube.com/watch?v=fVtUmhc0G5E
https://www.youtube.com/watch?v=sTeoEFzVNSc
https://www.youtube.com/watch?v=RR7rQLUpjTI
https://www.youtube.com/watch?v=I-kE11fhfaQ