Developing LLM Applications

DeveloperGPT: A simple open-source LLM-powered terminal application

Anthony Luo

May 22, 2024

Overview

1 Background

LLMs, Foundation & Instruction-Tuned Models, Prompting

2 DeveloperGPT

LLM-powered developer productivity command line tool

3 Developing with LLMs

Using DeveloperGPT as a guiding example

Background | Large Language Models

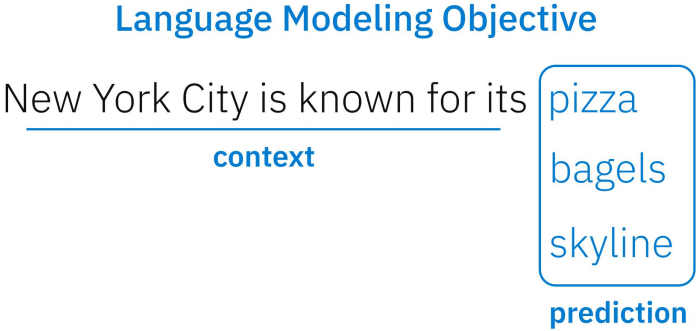
- **LLM =** Large Language Models = deep-learning models trained on large amounts of text
  - natural language understanding and generation
  - large = billions of parameters
- LLMs with **prompting** can be used for many applications they were not specifically trained on [1][2]
  - EX: reasoning, classification, translation, summarization, coding, etc.
- Some LLMs
  - OpenAI: *GPT-3.5, GPT-4*
  - Google: *Gemini, Gemma-7B*
  - Anthropic: *Claude 3*
  - Mistral AI: *Mistral-7B, Mixtral*
  - Meta: *LLAMA 3*
- Rapidly evolving field (e.g. Hugging Face Models)

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.

[2] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Background | Foundation LLMs

- "Base" LLMs trained (self-supervised learning) on large amount of unlabeled text
  - primary objective: **language modeling**
    - predict next word given prior words
  - byproduct: learn emergent abilities [2]
- Foundation models can be fine-tuned for a "wide-range of downstream tasks" [3]
  - **fine-tuning** = further training on domain specific information or tasks
- Foundation Models: BERT, BLOOM, GPT, etc.



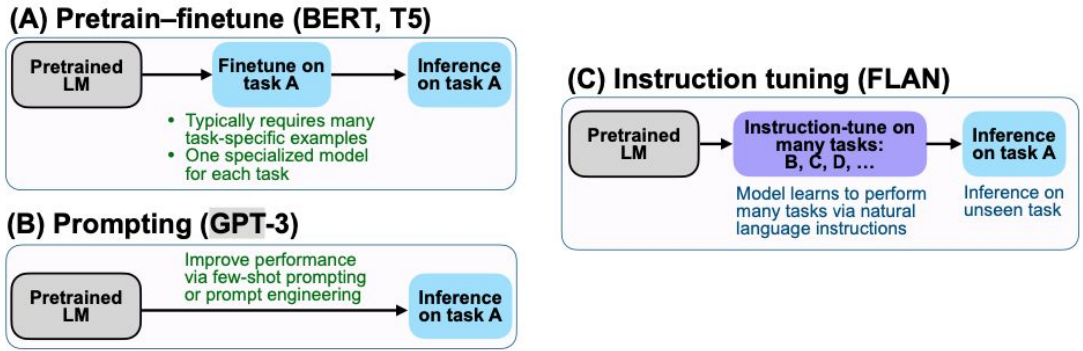
[2] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. arXiv preprint arXiv:2206.07682.

[3] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Background | Instruction-Tuned LLMs

- **Instruction-Tuned LLMs** = foundation LLMs further trained on (instruction, output) pairs [4]
  - improves ability of LLM to follow NLP instructions / tasks in general
  - often combined with Reinforcement Learning from Human Feedback (RLHF)
- **Fine-Tuning** = specific task, **Instruction-Tuning =** general NLP task following (type of fine-tuning)
  - requires significantly less data than training foundation model

Instruction Tuning vs. Pretrain-Finetune & Prompting from Wei et al. [4]



[4] Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., ... & Le, Q. V. (2021, October). Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Background | Prompting

- **Prompting** = representing many different tasks / problems as natural language input to LLM
  - many different prompting techniques, LLM-dependent
- **Prompting "Components"** [5]
  - *Objective / Instructions*: "what you want the model to achieve"
  - *Persona*: "who or what the model is acting as"
  - *Context*: additional information
  - *Response Format*: how the model should format its response
  - *Reasoning Steps*: how the model should think / explain its reasoning
  - *Few-Shot Examples*: task (input, output) response pairs
- Prompting can improve LLM performance for more complex or specific tasks [1][2]

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.

[2] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

[5] Google. (n.d.). *Prompting basics | Generative AI on Vertex AI | Google Cloud.* Google. https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design

Background | Zero to Few-Shot Prompting

Zero-Shot Prompting: asking model to perform task with no examples

> Identify the verbs in this sentence: The quick brown fox jumps over the lazy dog.

Few-Shot Prompting: asking model to perform tasks with examples provided [1]

- Can be necessary for more complex or specific tasks or for foundation models [4]
- “Learning from analogy” [6] - reinforce task instruction explicitly and implicitly
- Is NOT updating/training/tuning the LLM - just leveraging latent abilities of the LLM

> Identify nouns and verbs in the sentence and format the response in JSON.
>
> User: The quick brown fox jumped over the lazy dog while running away.
> Assistant: {“nouns”: [“fox”, “dog”], “verbs”: [“jump”, “run”]}
>
> User: She sold seashells by the seashore.
> Assistant: {“nouns”: [“seashells”, “seashore”], “verbs”: [“sold”]}

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.

[4] Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., ... & Le, Q. V. (2021, October). Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

[6] Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., ... & Sui, Z. (2022). A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

2 DeveloperGPT

LLM-powered developer productivity command line tool

DeveloperGPT | Overview

- LLM-powered open-source developer productivity terminal application
- **Free & open-source** alternative to Github CoPilot in the CLI (closed source, subscription product)

1 Natural Language Terminal Commands

Find all Python files in ~/Documents modified in the past day →

```
find /Users/anthony/Documents
    -mtime -1 -name '*.py'
```

2 In Terminal Chat

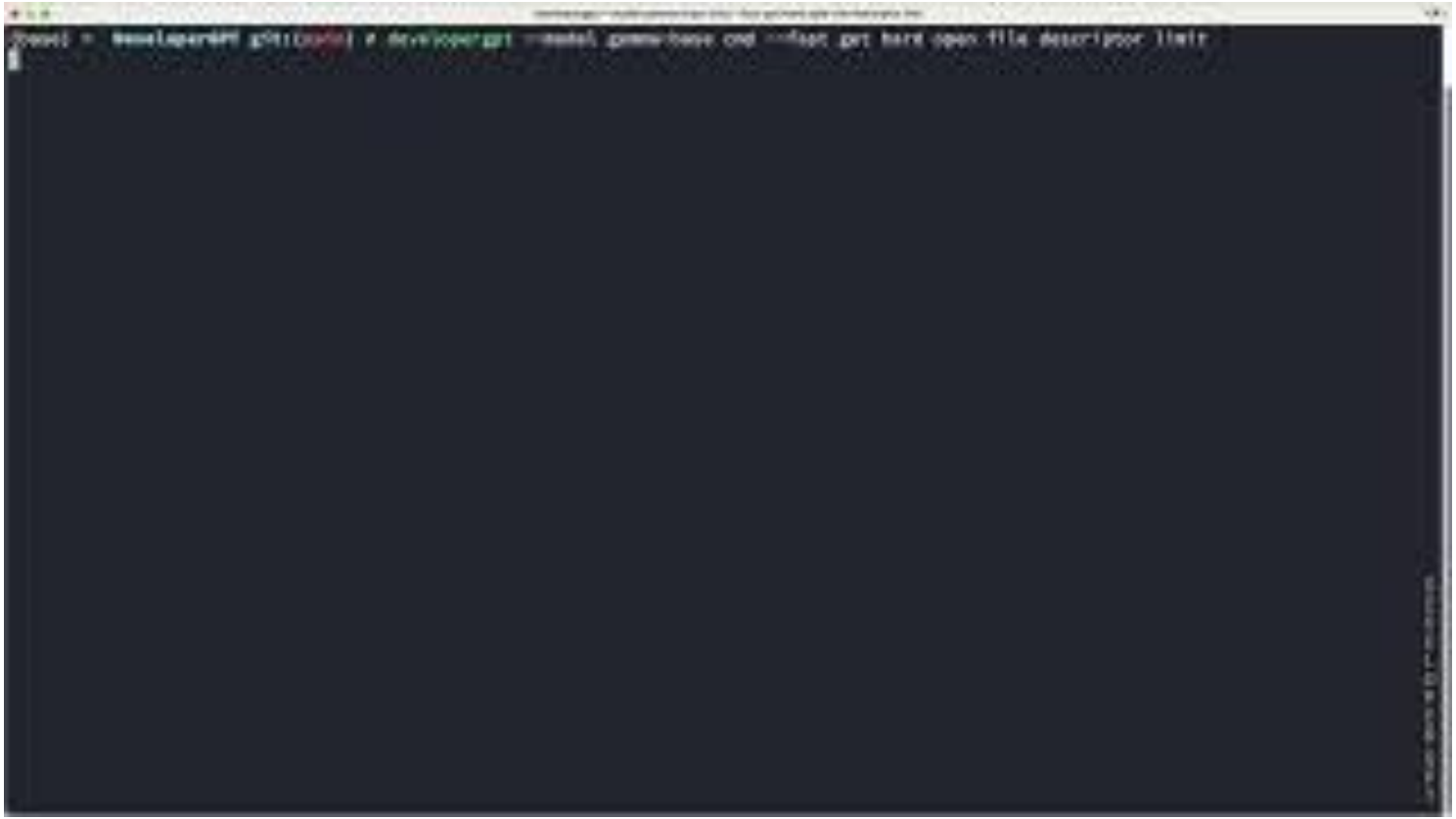
What are mixins in Python →

"In Python, mixins are a way to reuse code in multiple classes by combining them in a modular way....."

DeveloperGPT | Supported LLMs

LLMs	Source	Details
Gemini (default)	*Gemini 1.0 Pro, Gemini 1.5 Flash*	Free (up to 15 req/min), Google AI API Key Required
GPT-3.5, GPT-4	*GPT-3.5-Turbo, GPT-4-Turbo*	Pay-Per-Usage, OpenAI API Key Required
Haiku, Sonnet	*Anthropic Claude 3*	Pay-Per-Usage, Anthropic API Key Required
Mistral	*Mistral-7B-Instruct*	Free, Open LLM, Hugging Face Inference API
Zephyr	*Zephyr-7B-Beta*	Free, Open LLM, Hugging Face Inference API
Gemma, Gemma-Base	*Gemma-1.1-7B-Instruct* , *Gemma-7B-Base*	Free, Open LLM, Hugging Face Inference API
Mistral-Q6, Mistral-Q4	*Quantized GGUF Mistral-7B-Instruct*	Free, Open LLM, Offline, On-Device (llama.cpp)

DeveloperGPT | Demo



DeveloperGPT | Evaluation

Top@1 Accuracy on 85 Natural Language Command Requests [A]

95%

DeveloperGPT

Closed LLM, Pay-Per-Use

GPT-3.5

Regular Mode

84%

DeveloperGPT

Closed LLM, Free

Gemini 1.0 Pro

Fast Mode

67%

DeveloperGPT

Open LLM, Free

Gemma-7B-Base* or *Mistral-7B

Regular Mode

78%

Github CoPilot in the CLI

Closed LLM

Paid Subscription

v1.0.1 (03-22-2024)

DeveloperGPT | 1 of 85 Evaluation Requests

Delete all local git branches except the current one

✓
```
git branch | grep -v "$(git rev-parse --abbrev-ref HEAD)" | xargs git branch -D
```

✗
```
git branch --all | grep -v '*' | xargs git branch -D
```

✗
```
git branch | grep -v '(master|main)' | xargs git branch -D
```

DeveloperGPT | Takeaways

- Install from PyPI: https://pypi.org/project/developergpt/
  - `pip install -U developergpt`
- Contribute on Github: https://github.com/luo-anthony/DeveloperGPT
- NOTE: you should verify all commands provided by DeveloperGPT before running on your machine

Choosing an LLM: `--model [llm]` flag

- **Best Accuracy:** GPT-3.5*
  - `developergpt --model gpt35 cmd [your request here]`
- **Best Free:** Gemini 1.0 Pro (default)
  - `developergpt cmd [your request here]`
- **Offline, On-Device:** Quantized Mistral7B-Instruct
  - `developergpt --model mistral-q6 cmd [your request here]`

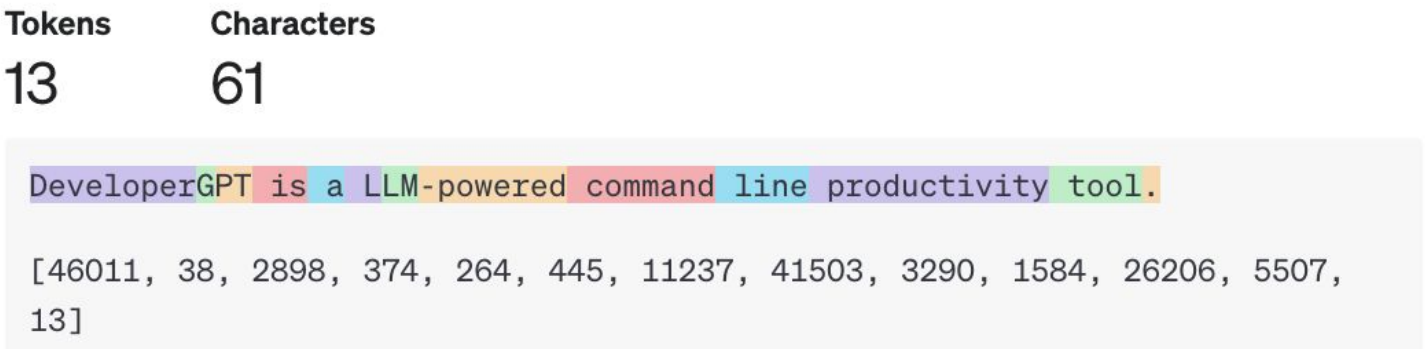
*average cost for DeveloperGPT with GPT-3.5 is < 0.0001 cents per query

3 Developing with LLMs

Using DeveloperGPT as a guiding example

Developing with LLMs | Development Basics

- **Tokenization** = transforming text into token representation for input to LLM
  - depends on the LLM used
- **Context Window** = number of tokens a given LLM can accept as input at once
  - 1 token = ~4 characters of English (100 tokens ~= 75 words) [7]
- Use library such as tiktoken for tokenization



Tokenizer used in GPT-3.5 & GPT-4 [7]

[7] OpenAI. (n.d.). *Tokenizer*. OpenAI. https://platform.openai.com/tokenizer

Developing with LLMs | LLMs are Stateless

LLMs are stateless - continuous conversation history is a software-provided illusion

- **Stateless** = LLMs do not retain state or memory between sessions
  - LLM generates response based on only the **current input** and model pre-trained knowledge
  - LLM does not have knowledge of previous inputs or outputs
- If you need LLM to be aware of history, the **history needs to be provided to the LLM as part of each input by your application**
  - conversation history & context must be maintained/store by software interface to LLM
- NOTE: while LLMs are fundamentally stateless, LLM API inputs/outputs may still be used for training or retained by the LLM provider

Developing with LLMs | LLMs are Stateless

User-LLM Conversation

What is the difference between x86 and ARM?

x86 and ARM architectures differ primarily in their design philosophies and application use cases. x86, used predominantly in desktop and laptop computers, is based on a Complex Instruction Set Computing (CISC)...

Can you compare these in the context of HPC?

In the realm of High-Performance Computing (HPC), x86 architecture has long held sway due to its widespread adoption, robust ecosystem, and high computational performance, particularly in tasks benefiting from parallel processing and large memory bandwidth...

Input to LLM

User: What is the difference between x86 and ARM?

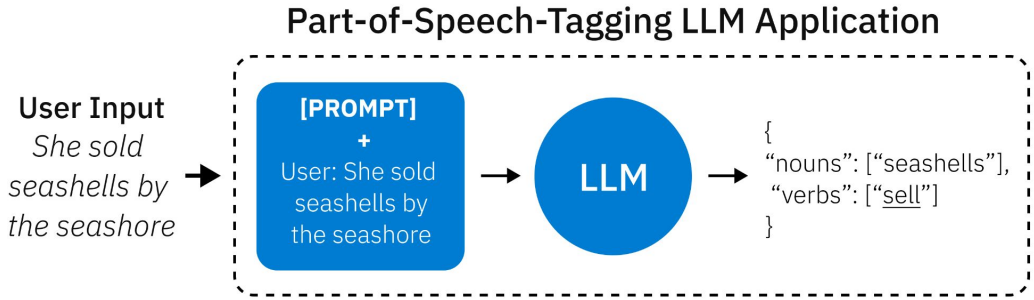
[PREVIOUS CONVERSATION HISTORY]
User: What is the difference between x86 and ARM?

Assistant: x86 and ARM architectures differ primarily in their design philosophies and application use cases. x86, used predominantly in desktop and laptop computers, is based on a Complex Instruction Set Computing (CISC)...

[NEW]
User: Can you compare these in the context of HPC?

Developing with LLMs | Prompting

- LLM + Prompting = Fully Customizable, Highly Adaptable API
- Can “offload” logic/complexity of application onto the LLM
- Need to handle downsides of LLMs
  - unpredictable outputs, hallucinations, deviations



[PROMPT]

Identify nouns and verbs in a valid sentence and return the following JSON format: {“nouns”: [“n1”, “n2”, ...], “verbs”: [“v1”, “v2”, ...]}

Convert all verbs to their lemmatized forms.

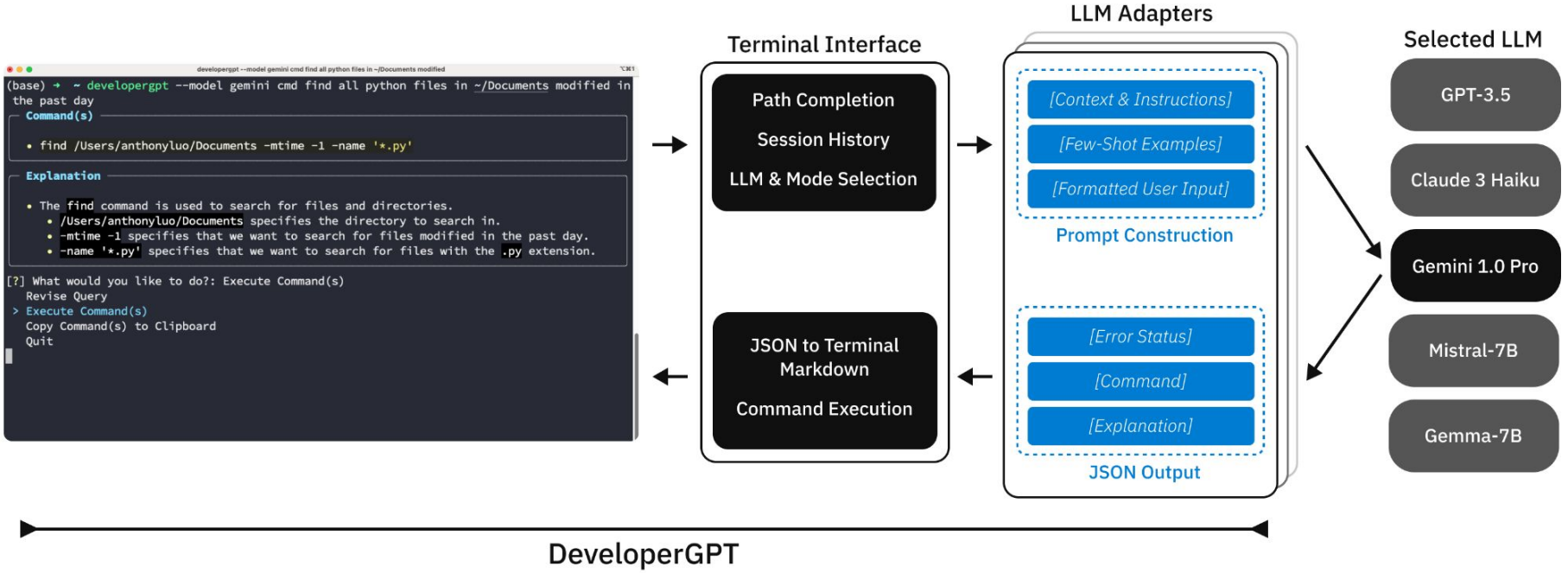
If the sentence is not valid, return:
{“error”: 1, “explanation”: “reason why the sentence is invalid”}

[FEW-SHOT EXAMPLES]

User: The quick brown fox jumped over the lazy dog while running away.
Assistant: {“nouns”: [“fox”, “dog”], “verbs”: [“jump”, “run”]}

User: qwertyuiop
Assistant: {“error”: 1, “explanation”: “qwertuiop is not a valid word, and does not form a proper sentence”}

DeveloperGPT | Thin Wrapper around LLM



Developing with LLMs | DeveloperGPT Prompts

Context + Instructions for **Instruction-Tuned Models**

```
Provide the appropriate command-line commands that can be executed on a {platform} machine for the user request: "{user_request}".
If the request is possible, please provide commands that can be executed in the command line and do not require a GUI.
Today's date/time is {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}.
Do not include commands that require a yes/no response.
For each command, explain the command and any arguments used.
Try to find the simplest command(s) that can be used to execute the request.
```

Response Format for **Instruction-Tuned Models**

```
If the request is valid, format each command output in the following JSON format: {cmd_format}
If the request is invalid, please return the following JSON format: {invalid_format}
```

Developing with LLMs | DeveloperGPT Prompts

Regular-Mode JSON Response Format

```
{"input": "<user input>",
    "error": 0,
    "commands": [
        {
            "seq": <Order of Command>,
            "cmd_to_execute": "<commands and arguments to execute>",
            "cmd_explanations": ["<explanation of command 1>", "<explanation of command 2>", ...],
            "arg_explanations": {"<arg1>": "<explanation of arg1>"...}
        },
   ]
}
```

Fast-Mode JSON Response Format

```
{"commands": ["<commands and arguments to execute>", "<commands and arguments to execute>", ...]}
```

Developing with LLMs | DeveloperGPT Prompts

Few Shot Examples for **Foundation & Instruction-Tuned Models**

```
User: "search ~/Documents directory for any .py file that begins with 'test'"
Assistant: {"input": "search the ~/Documents/ directory for any .py file that begins with 'test'",
            "error" : 0,
            "commands": [{"seq": 1,
                          "cmd_to_execute": "find ~/Documents/ -name 'test*.py'",
                          "cmd_explanations": ["`find` is used to list files."],
                          "arg_explanations": {
                                "~/Documents": "specifies the folder to search in.",
                                "-name 'test*.py'": "specifies that we want to search for files
starting with `test` and ending with `.py`."}...
```

Example of Invalid Request

```
User: "the quick brown fox jumped over"
Assistant: {"input": "the quick brown fox jumped over", "error": 1}
```

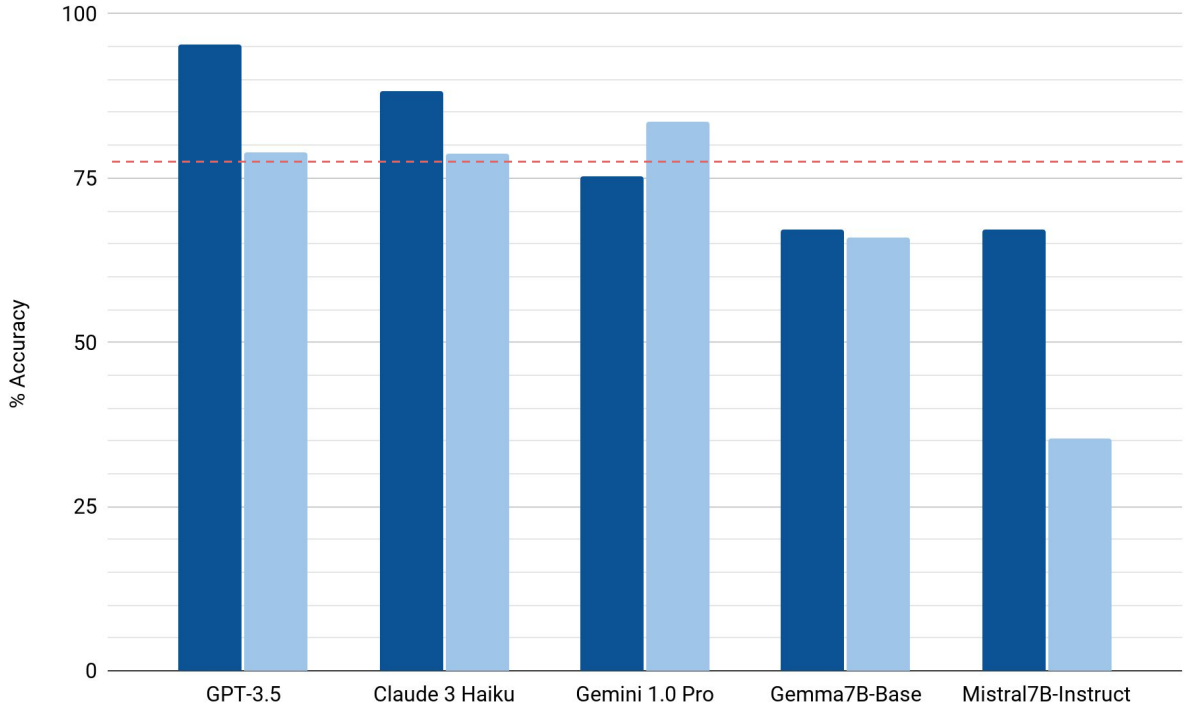
Developing with LLMs | DeveloperGPT Prompts

Prompt for **Foundation Models**

The following is a software development command line system that allows a user to get the command(s) to execute their request in natural language.
The system gives the user a series of commands to be executed for the given platform in JSON format along with explanations.
User: Provide appropriate command-line commands that can be executed on a macOS-13.3.1-x86-64bit machine to: "search ~/Documents directory for any .py file that begins with 'test'".
Assistant: {"input": "search the ~/Documents/ directory...}
User: Provide ... to: "list all processes using more than 50 MB of memory".
Assistant: {"input": "list all processes using more than 50...}
User: Provide … on a **{platform}** machine to: "**{user_request}**".
Assistant: [LLM completes text here]

DeveloperGPT | Regular vs. Fast Mode Prompt

Top@1 Accuracy on 85 Natural Language Command Requests [A]



Regular Mode

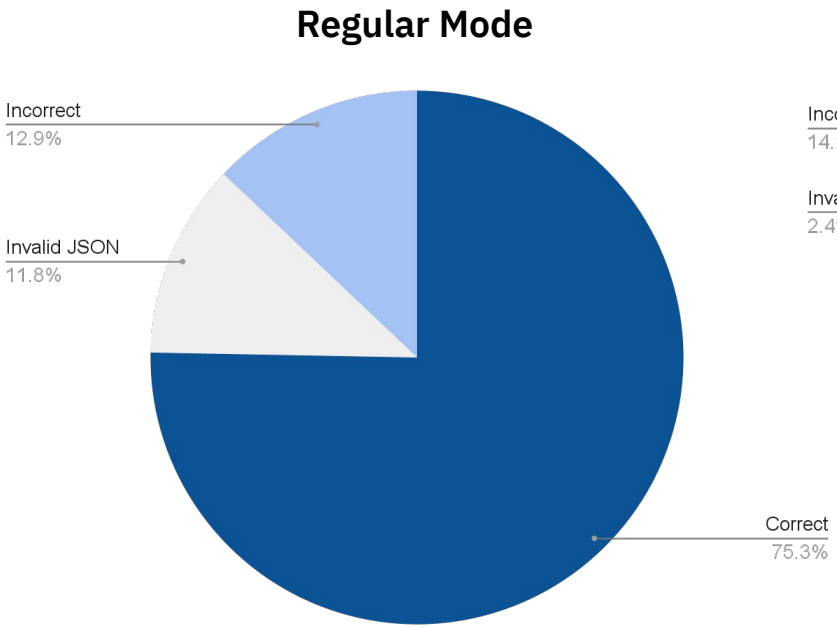
LLM JSON Output with Command & Argument Explanations + Sequencing

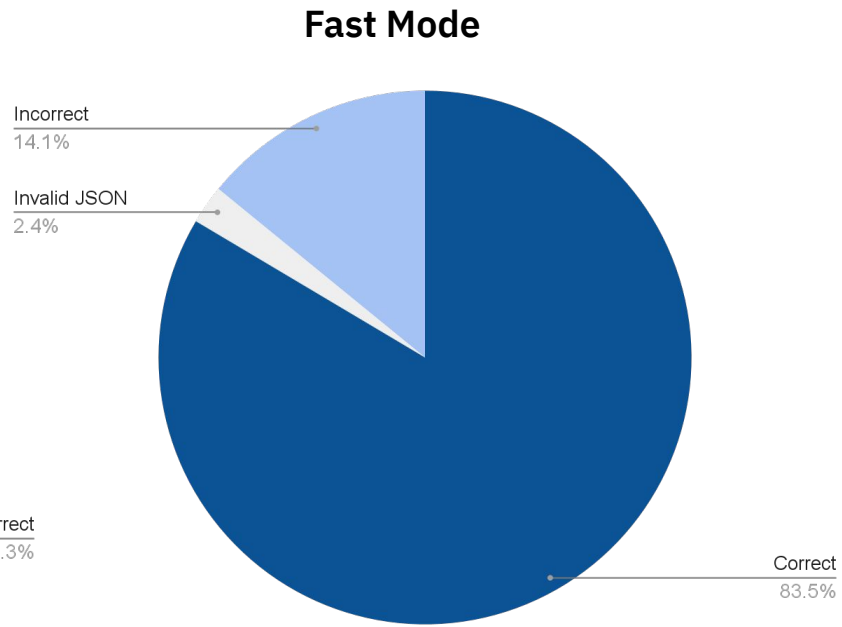
Fast Mode

LLM JSON Output with Commands Only

DeveloperGPT | Regular vs. Fast Mode Prompts

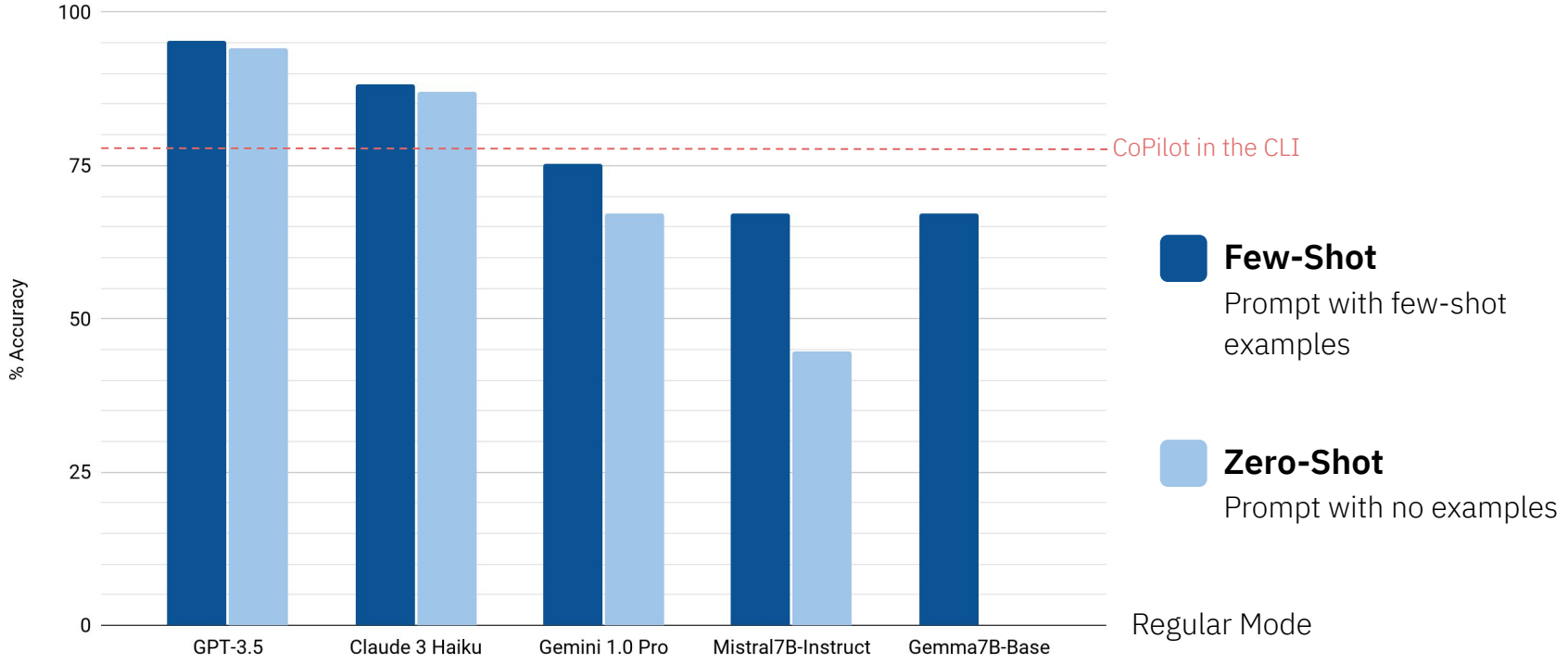
Gemini 1.0 Pro Natural Language Command Request Breakdown





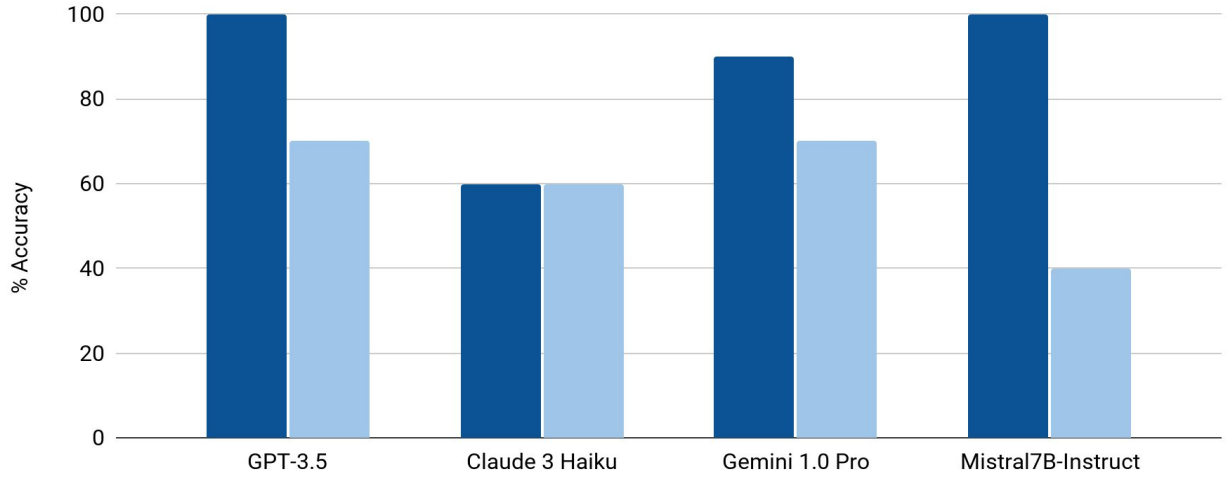
DeveloperGPT | Few-Shot vs. Zero-Shot Prompts

Top@1 Accuracy on 85 Natural Language Command Requests [A]



DeveloperGPT | Few-Shot vs. Zero-Shot Prompts

Top@1 Accuracy for 10 Invalid Command Requests [B]



Few-Shot
Prompt with few-shot examples

Zero-Shot
Prompt with no examples

Few-Shot Example of Invalid Request

User: `"the quick brown fox jumped over"`

Assistant: `{"input": "the quick brown fox jumped over", "error": 1}`

DeveloperGPT | Few-Shot vs. Zero-Shot Prompts

Claude 3 Haiku LLM Output with Few-Shot Prompt

```
{"input": "find all python files less than 5kB in size in /Users/anthonyluo/Desktop/demo that were modified in the past day",
 "error": 0,
 "commands": [{...}]
}
```

Claude 3 Haiku LLM Output with Zero-Shot Prompt

Here are the appropriate command-line commands to execute the user request on a macOS-14.4.1-arm64-arm-64bit machine:

```
{"input": "find all python files less than 5kB in size in /Users/anthonyluo/Desktop/demo that were modified in the past day",
 "error": 0,
 "commands": [{...}]
}
```

The command provided will search the '/Users/anthonyluo/Desktop/demo' directory for Python files (*.py) that are less than 5 kilobytes in size and were modified within the last 24 hours. The output of this command will be a list of matching files.

Developing with LLMs | Basic Prompting Takeaways

- A good prompt can improve LLM performance & structured output
  - important to evaluate and iterate on prompts when developing with LLMs
- Prompts depend on the size and type (instruction vs. foundation) of LLM
- Use well-curated few-shot examples in prompts
  - helps LLM recognize invalid input and return error output expected by application
  - generally needed when using foundation model
  - smaller LLMs may benefit more from few-shot examples than larger LLMs
- More prompting / more few-shot examples is not necessarily better
  - longer prompts = slower inference speed & higher cost
- Prompt engineering guides: OpenAI, Anthropic, Google, etc.

Developing with LLMs | Structured Output

- Structured & predictable LLM outputs are essential for good LLM-application integration
- Many techniques and frameworks beyond prompting
  - lower model temperature (or Top-P) = less randomness, more deterministic outputs
  - JSON response format in LLM API [8]
  - structured output frameworks: Outlines, LlamaIndex Pydantic Program, GGML BNF, OpenAI Function Calling, etc.
- DeveloperGPT achieved largely consistent structured output with prompting + lower temperature

[8] OpenAI. (n.d.). *Chat create.* OpenAI API Reference. https://platform.openai.com/docs/api-reference/chat/create#chat-create-response_format

Developing with LLMs | Choosing an LLM

- **LLM Performance** - Benchmarks, Real-World Evaluation
- **LLM Technical Specs** - Context Window, Inference Speed, Instruction-Tuned vs. Foundation [9]
- **Cost** - Cost per million input & output tokens for given LLM

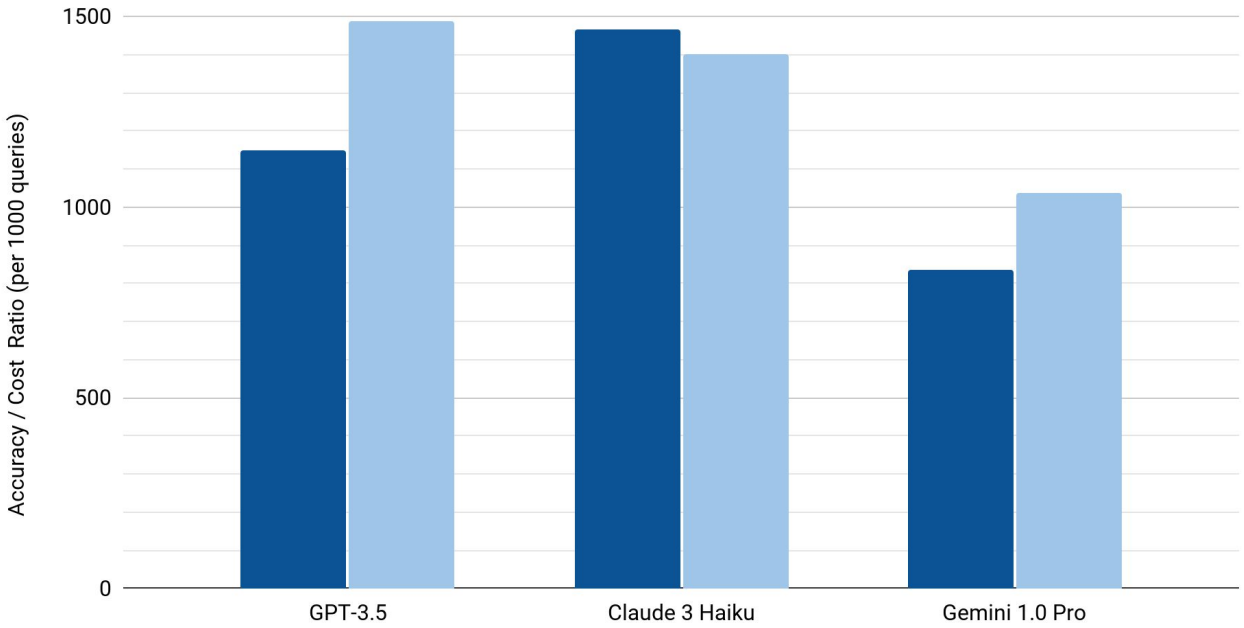
Closed vs. Open LLM

- **Closed LLM**
  - best top-end performance, more robust/reliable API, better libraries and documentation
  - more expensive, less customizable, beholden to corporation
- **Open LLM**
  - almost-infinite flexibility and customization, cheaper to use (or free), community support
  - worse top-end performance, less robust APIs, steeper learning curve

[9] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics, 12*, 157-173.

DeveloperGPT | Trade Offs

Accuracy / Cost Ratio in Regular Mode



Few-Shot
Prompt with few-shot examples

Zero-Shot
Prompt with no examples

Calculated using LLM API costs as of May 21, 2024

Developing with LLMs | Levels

- **Level 0: Talking to LLM**
- **Level 1: Prompting** ← DeveloperGPT
  - prompt frameworks: MedPrompt [10], DSpy [11], etc.
  - prompt techniques: chain-of-thought [12], least-to-most [13], etc.
- **Level 2: Augmentation**
  - LLM is augmented with additional information, memory, tools, planning, etc.
  - LLM is NOT updated / fine-tuned
- **Level 3: Fine-Tuning**
  - pre-trained LLM is additionally trained for specific domain or application
- **Level 4: Training Foundation Model**

[10] Nori, H., Lee, Y. T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., ... & Horvitz, E. (2023). Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*.
[11] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., ... & Potts, C. (2023). Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.
[12] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35, 24824-24837.
[13] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., ... & Chi, E. (2022). Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Questions?

Feel free to contact me on linkedin.com/in/luo-anthony

References

[1] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33, 1877-1901.

[2] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., ... & Fedus, W. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

[3] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... & Liang, P. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

[4] Wei, J., Bosma, M., Zhao, V., Guu, K., Yu, A. W., Lester, B., ... & Le, Q. V. (2021, October). Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

[5] Google. (n.d.). *Prompting basics | Generative AI on Vertex AI | Google Cloud*. Google. https://cloud.google.com/vertex-ai/generative-ai/docs/learn/prompts/introduction-prompt-design

[6] Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., ... & Sui, Z. (2022). A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

[7] OpenAI. (n.d.). *Tokenizer*. OpenAI. https://platform.openai.com/tokenizer

[8] OpenAI. (n.d.). *Chat create.* OpenAI API Reference. https://platform.openai.com/docs/api-reference/chat/create#chat-create-response_format

[9] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics, 12*, 157-173.

[10] Nori, H., Lee, Y. T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., ... & Horvitz, E. (2023). Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*.

[11] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., ... & Potts, C. (2023). Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*.

[12] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35, 24824-24837.

[13] Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., ... & Chi, E. (2022). Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Appendix

[A] 85 Natural Language Terminal Command Requests

https://github.com/luo-anthony/DeveloperGPT/blob/evaluation_v2/evaluation/85_command_requests.txt

[B] 10 Invalid Command Requests

https://github.com/luo-anthony/DeveloperGPT/blob/evaluation_v2/evaluation/10_invalid_requests.txt