Introduction to prompt engineering with GitHub Copilot

Introduction

Prompt engineering is how you tell GitHub Copilot what you need. The quality of the code it gives back depends on how clear and accurate your prompts are.

At the end of the module, you'll understand:

- **Prompt Engineering**: Learn principles and best practices for crafting effective prompts to optimize GitHub Copilot's performance.
- **Processing Flow**: Understand how GitHub Copilot processes user prompts to generate context-aware responses and code suggestions.
- **Data Flow**: Explore the data flow for code suggestions and chat, including secure transmission and content filtering.
- **Role of LLMs**: Discover the role of Large Language Models (LLMs) in GitHub Copilot and how they influence prompting and responses.

Prompt engineering foundations and best practices

What is Prompt Engineering?

- Craft clear instructions to guide AI systems like GitHub Copilot.
- Ensures code is syntactically, functionally, and contextually correct.
- Generates context-appropriate code tailored to project needs.

Principles of Prompt Engineering

- Focus on a single, well-defined task or question for clarity.
- Provide explicit and detailed instructions for precise code suggestions.
- Keep prompts concise to maintain clarity without overloading the AI.

Best Practices in Prompt Engineering

- Use descriptive filenames and keep related files open for rich context.
- **Iterate**: Refine prompts based on feedback to improve AI responses.
- **Contextualize**: Provide relevant background information to enhance code suggestions.

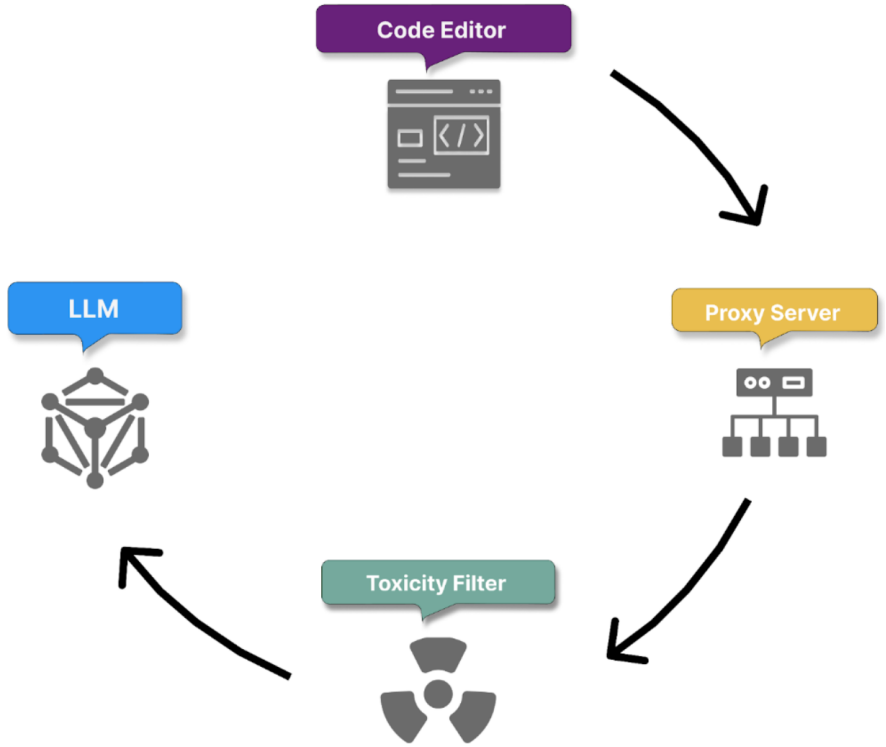
Advanced Strategies

- **Layered Prompts**: Break complex tasks into smaller, manageable prompts.
- **Feedback Loop**: Continuously evaluate and adjust prompts for optimal performance.
- **Collaborative Input**: Involve team members to ensure comprehensive and effective prompts.

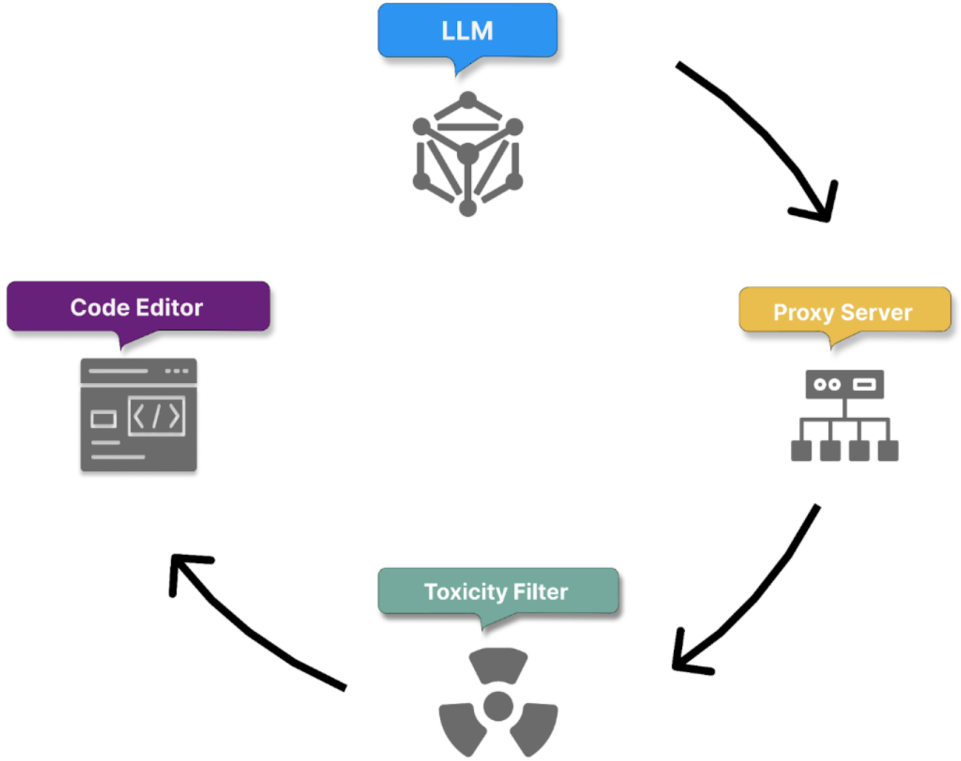
GitHub Copilot user prompt process flow

GitHub Copilot receives prompts and returns code suggestions or responses in its data flow. This process suggests an inbound and outbound flow.

Inbound Flow



Outbound Flow



GitHub Copilot data

Data Handling for Code Suggestions

- **No Retention**: GitHub Copilot in the code editor discards prompts once a suggestion is returned.
- **Opt-Out Option**: Individual subscribers can opt-out of sharing their prompts for model fine-tuning.

Data Handling for Copilot Chat

- **Response Formatting**: Formats responses for optimal presentation, highlighting code snippets.
- **User Engagement**: Allows follow-up questions and maintains conversation history for context.
- **Data Retention**: Retains prompts and suggestions for 28 days outside the code editor.

Prompt Types Supported by Copilot Chat

- **Direct Questions**: Ask specific coding questions or troubleshoot issues.
- **Code-Related Requests**: Request code generation, modification, or explanations.
- **Open-Ended Queries**: Explore coding concepts or seek general guidance.

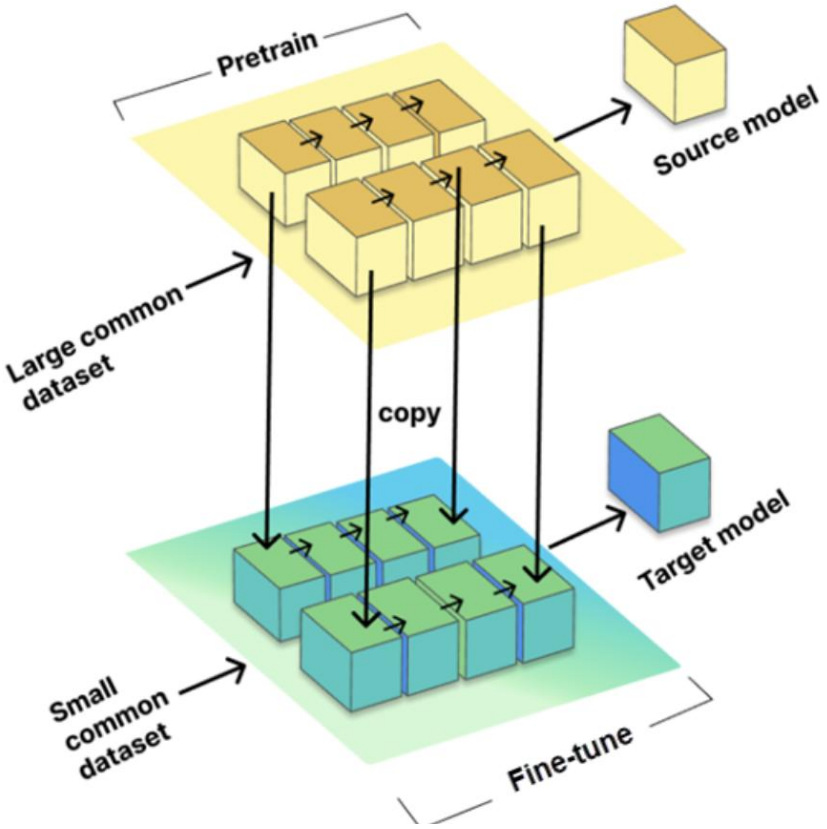
Contextual Prompts

- **Code Snippets**: Provide snippets for tailored assistance.
- **Scenario Descriptions**: Describe coding scenarios for specific help.

GitHub Copilot Large Language Models (LLMs)

Fine-tuning LLMs involves training the model on a smaller, task-specific dataset, known as the target dataset, while using the knowledge and parameters gained from the source model.

- **Volume of Training Data**: LLMs are trained on vast amounts of text from diverse sources, providing a broad understanding of language and context.
- **Contextual Understanding**: They generate contextually relevant and coherent text, making meaningful contributions to various forms of communication.
- **Machine Learning Integration**: LLMs are neural networks with millions or billions of parameters, fine-tuned to understand and predict text effectively.
- **Versatility**: These models can be tailored for specialized tasks, making them highly versatile across different domains and languages.



Summary

Now that you've finished this module, you should be able to describe:

- **Prompt Engineering**: Learn principles and best practices for crafting effective prompts to optimize GitHub Copilot's performance.
- **Processing Flow**: Understand how GitHub Copilot processes user prompts to generate context-aware responses and code suggestions.
- **Data Flow**: Explore the data flow for code suggestions and chat, including secure transmission and content filtering.
- **Role of LLMs**: Discover the role of Large Language Models (LLMs) in GitHub Copilot and how they influence prompting and responses.

Microsoft

Thank you.