Code-survey

A LLM-Powered Approach to Understanding Large Codebases

By Yusheng Zheng

https://github.com/yunwei37

https://github.com/eunomia-bpf

Agenda

- What is "Understanding Codebases" means?
- LLMs struggle to answer questions about large-scale codebases
- Another approach to gain insights into repo
- Example: eBPF in Linux kernel
- Best Practices
- Limitations
- What's next?

Can LLM understand code?

We know LLM can help find bugs and write code.

Understand code is not only for development and fix bugs. We are trying to answer more high level questions:

- How do new features affect stability and performance?
- What are the stages in a component’s lifecycle?
- How have specific features changed over time?
- Which components/files are most prone to bugs?

Why need to understand these?

Help design software

Do bug study and find root cause

Improved Code Review process

Enhanced Software Stability

Aid in design new Debugging tools

Support for Newcomers in OSS projects

Why LLM?

It's like Empirical Studies in Software Engineering

- **Limitations of Traditional Methods:**
  - Static analysis and manual code reviews are time-consuming and incomplete
- **Challenges with Unstructured Data:**
  - Valuable insights hidden in commits, emails, and discussions

Traditional methods are Limited by scale, biases, and subjectivity

Problem: hard for large codebases

- **In-Context Learning**
  - Using examples during inference
  - Limited by context, struggles with complex projects
- **RAG (Retrieval-Augmented Generation)**
  - Combining LLMs with external knowledge sources
  - Relies on accurate retrieval
- **Fine-Tuning**
  - Adapting models with specific datasets
  - Costly and time-consuming, risks overfitting on limited data

They are all hard to answer high level design questions across multiple files.

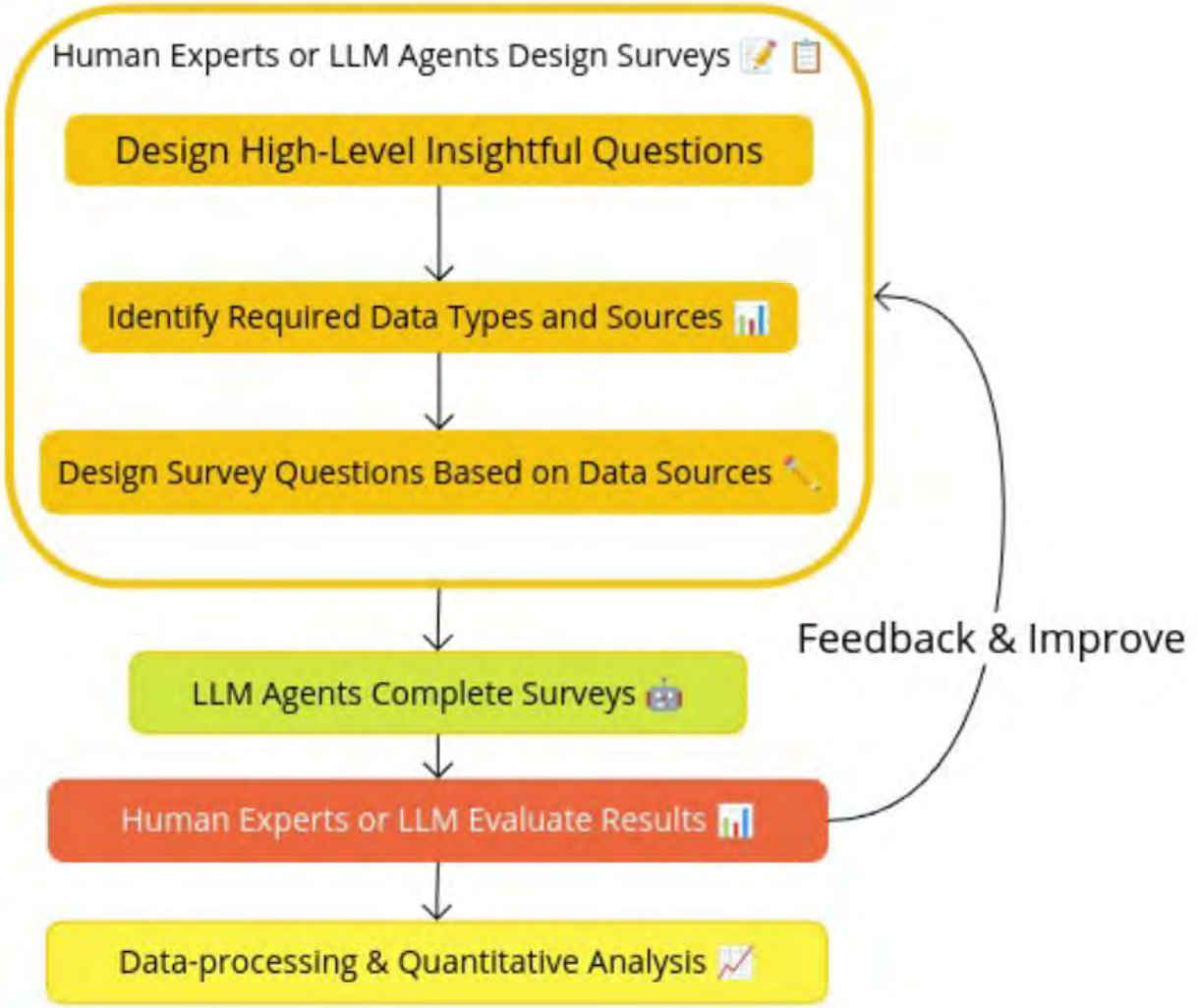
Code-survey: another approach

An LLM-driven methodology for analyzing codebases

"What if we could ask developers to take a survey about everything they contributed during development?"

- Treat LLMs as human participants in surveys
- Commits, emails, everything ➔ Structured datasets
- Quantitative insights into software evolution

How it works


Human Experts or LLM Agents Design Surveys 📝 📋
Design High-Level Insightful Questions
Identify Required Data Types and Sources 📊
Design Survey Questions Based on Data Sources ✏️
LLM Agents Complete Surveys 🤖
Human Experts or LLM Evaluate Results 📊
Data-processing & Quantitative Analysis 📈
Feedback & Improve

Example: survey define

```yaml
title: "Commit Classification Survey"
description: "A survey about the commit in Linux eBPF, to help better understand the design and evolution of bpf subsystem. For choice, try to be as specific as possible based on the commit message and code changes. If the commit message is not clear or does not provide enough information, you can choose the 'I'm not sure' option."
hint: "For example, when seems not related to eBPF, confirm it's a rare cases really has nothing to do with eBPF in all it's contents, such as btrfs or misspelled commit message. Do not tag subsystem changes related to eBPF as not."
questions:
...
- id: commit_classification
  type: single_choice
  question: "What may be the main type of the commit?"
  choices:
    - value: A bug fix. It primarily resolves a bug or issue in the code.
    - value: A new feature. It adds a new capability or feature that was not previously present.
    - value: A performance optimization. It improves the performance of existing code such as reducing latency or improving throughput.
    - value: A cleanup or refactoring in the code. It involves changes to improve code readability maintainability or structure without changing its functionality.
    - value: A documentation change or typo fix. It only involves changes to documentation files or fixes a typographical error.
    - value: A test case or test infrastructure change. It adds or modifies test cases test scripts or testing infrastructure.
    - value: A build system or CI/CD change. It affects the build process continuous integration or deployment pipelines.
    - value: A security fix. It resolves a security vulnerability or strengthens security measures.
    - value: It's like a merge commit. It merges changes from another branch or repository.
    - value: It's other type of commit. It does not fit into any of the categories listed above.
    - value: I'm not sure about the type of the commit. The nature of It is unclear or uncertain.
```

Example: Prompt

```
--------------------
Processing Commit 1/7346 - Commit ID: bc4f0548f683a3d53359cef15f088d2d5bb4bc39
Survey Title: Commit Classification Survey


Description: A survey about the commit in Linux eBPF, to help better understand the design and evolution of bpf subsystem. For (

Commit Details:
  Commit ID: bc4f0548f683a3d53359cef15f088d2d5bb4bc39
  Author Name: Yonghong Song
  Author Email: yhs@fb.com
  Commit Date: 1595363186
  Commit Message:
    bpf: Compute bpf_skc_to_*() helper socket btf ids at build time  Currently, socket types (struct tcp_sock, udp_sock, etc.) (
  Changed Files: include/linux/bpf.h |  4 ----; kernel/bpf/btf.c    |  1 -; net/core/filter.c   | 49 ++++++++++++++++++---------
  Parent Hashes: e4d9c2320716ea0e9ef59f503ddd8f253a642ddd
- Please provide a summary of It in one short sentence not longer than 30 words. Only output one sentence.

- Please extract no more than 3 keywords from the commit. Only output 3 keywords without any special characters.

- What may be the main type of the commit?

- What is the estimated complexity of implementing this commit considering file and line changes? Note that complexity is most (

- What major implementation component is modified by the commit? It's typically where the code changes happened.

- What major logic component is affected by the commit? Logic component is different from implementation component. it's typica

- What major eBPF use cases events or hooks in other subsystems may the commit relate to and be designed for? Note a lot of the
```

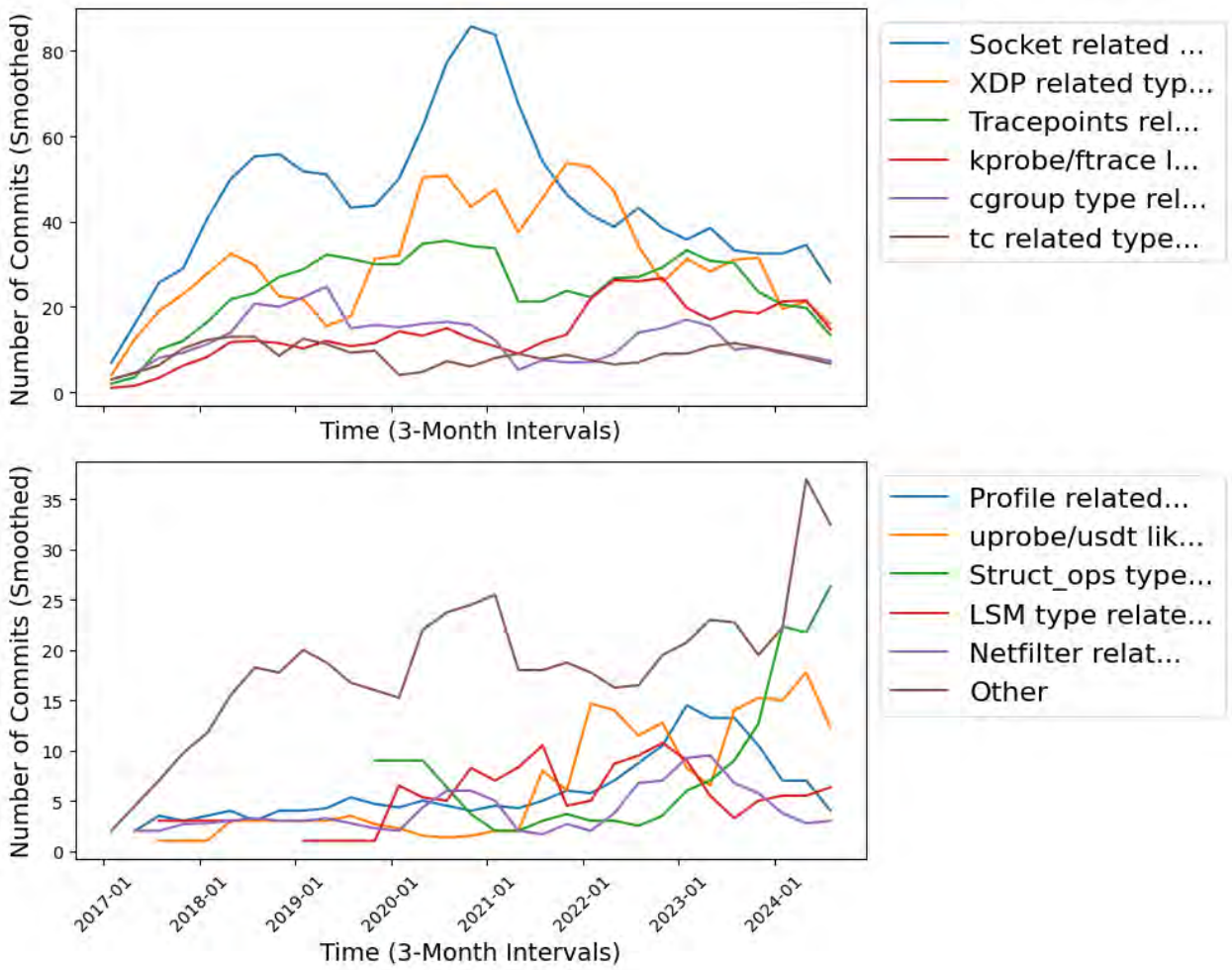
Case Study: Linux kernel eBPF

- **What is eBPF?**
  - Extended Berkeley Packet Filter
  - Allows running sandboxed programs in the kernel
- **Why eBPF and Linux?**
  - Let kernel Expert Confirm our findings (We are doing eBPF ourselves!)
  - Rapidly evolving subsystem in Linux
  - Complexity and critical functionalities
  - Linux are developed with emails and patches
- **Goals:**
  - Apply Code-Survey to analyze eBPF evolution

Features changed over time:

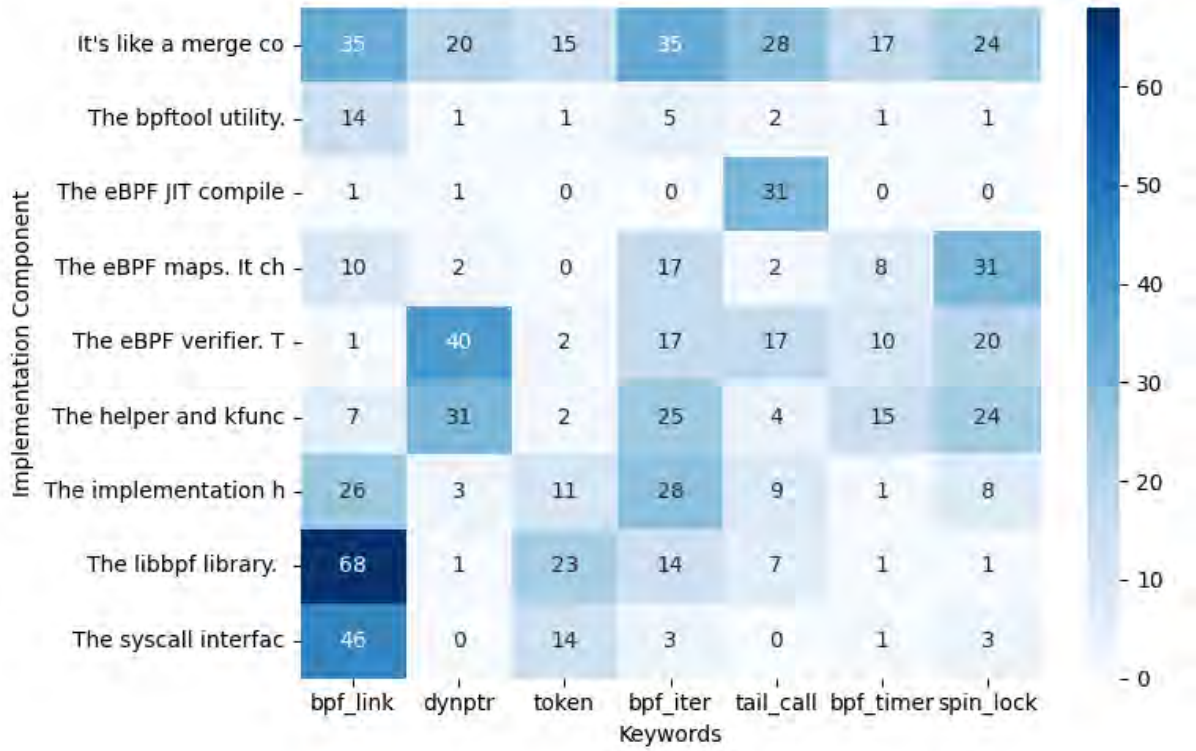
eBPF events development

Results for eBPF



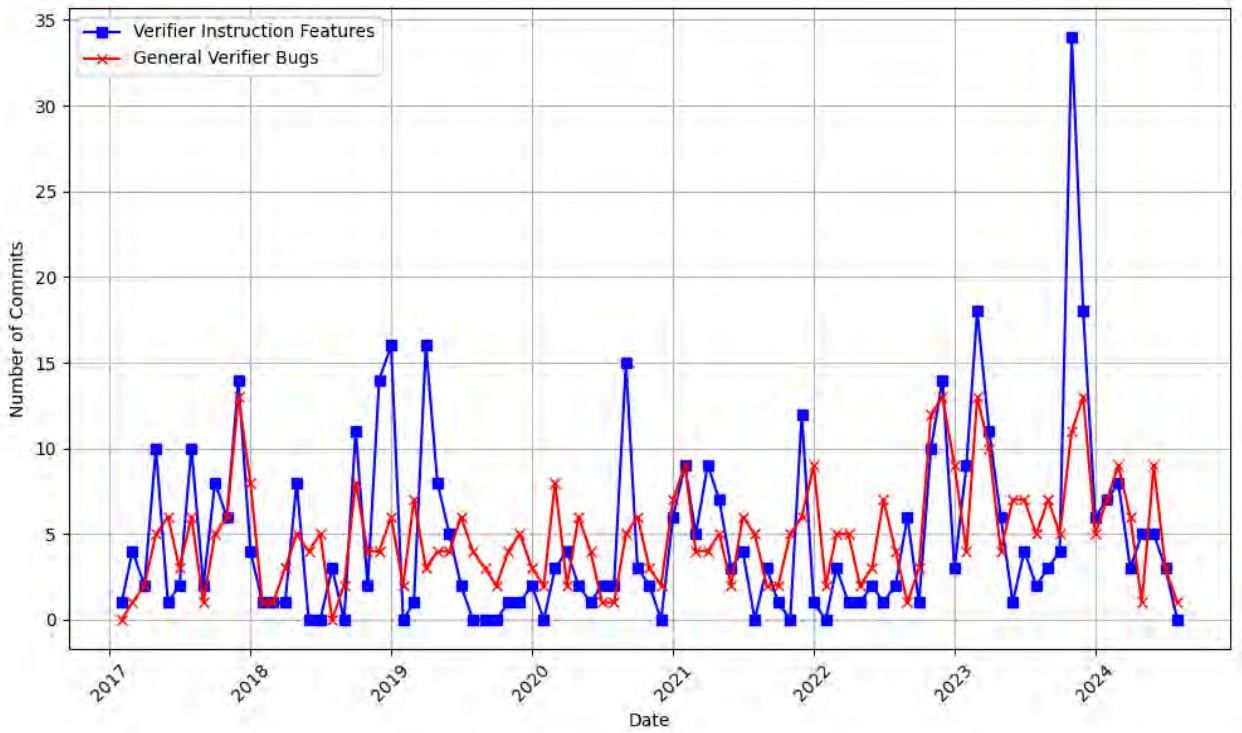
Feature-Component Interdependencies: which component need to modify when a feature is added

Results for eBPF



Does adding eBPF instructions lead to most bugs in eBPF verifier?

Results for eBPF



Detailed report can be found in Github and Arxiv paper!

Best Practices in Code-Survey

Guidelines:

- Use predefined tags and categories
- Implement As agent with dynamic workflows
- Allow "I'm not sure" responses
- Pilot testing and iterative refinement
- Ensure consistency and data validation
- Run multiple times to get average results

Goal:

- Improve reliability and accuracy of LLM-generated data

Limitations Now

- **LLM Mistakes:**

  Potential for misinterpretations and hallucinations
- **Dependence on Data Quality:**

  Incomplete data can lead to gaps
- **Need for Human Expertise:**

  Essential for survey design and validation

What's next?

Automation with less human effort: Refine processes with advanced Agents

Enhanced Evaluation: Build robust frameworks for validation and consistency checks

Performance Optimization: Leverage powerful models like O1 and design new workflow for higher accuracy and speed

Broader Application: Expand methodology to projects like Kubernetes, LLVM, and Apache

Thanks

- Github: https://github.com/eunomia-bpf/code-survey
- Arxiv: https://arxiv.org/abs/2410.01837