Contextual Augmented Multi-Model Programming (CAMP): A Local-Cloud Copilot Solution

Yuchen Wang, Shangxin Guo, Chee Wei Tan

Y. Wang, S. Guo and C. W. Tan, "From Code Generation to Software Testing: AI Copilot with Context-Based RAG," IEEE Software, 2025, in print doi: 10.1109/MS.2025.3549628.
Y. Wang, S. Guo and C. W. Tan, IEEE Conference on Artificial Intelligence, 2025
https://arxiv.org/abs/2410.15285

Introduction

Introducing **CAMP**: A Collaborative Multi-Model Copilot using Context-Aware Retrieval-Augmented Generation (RAG) to bridge cloud LLMs with local models.

- +12.5% over non-contextual generation
- +6.3% over baseline RAG
- Deployed as "Copilot for Xcode"
- Integrated with GitHub Copilot

Copilot for Xcode: exploring AI-assisted programming by prompting cloud-based large language models
CW Tan, S Guo, MF Wong, CN Hang, 2023, https://arxiv.org/abs/2307.14349
M. F. Wong and C. W. Tan, "Aligning Crowd-Sourced Human Feedback for Reinforcement Learning on Code Generation by Large Language Models," in IEEE Transactions on Big Data, 2024.

Motivation

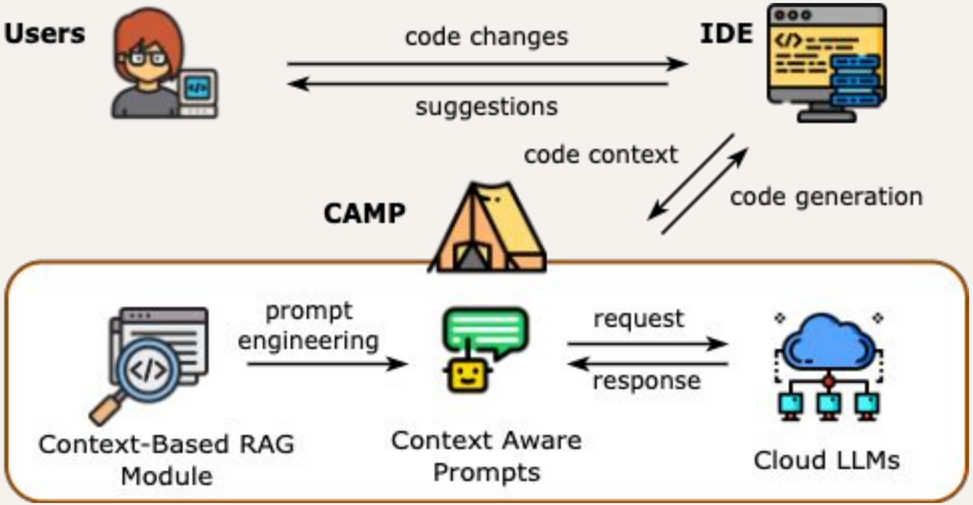
Challenge: Cloud-based LLMs are powerful for code generation, but lack local context. Locally integrated tools are adaptive, but limited in scope.

Local models: light weight, has code context

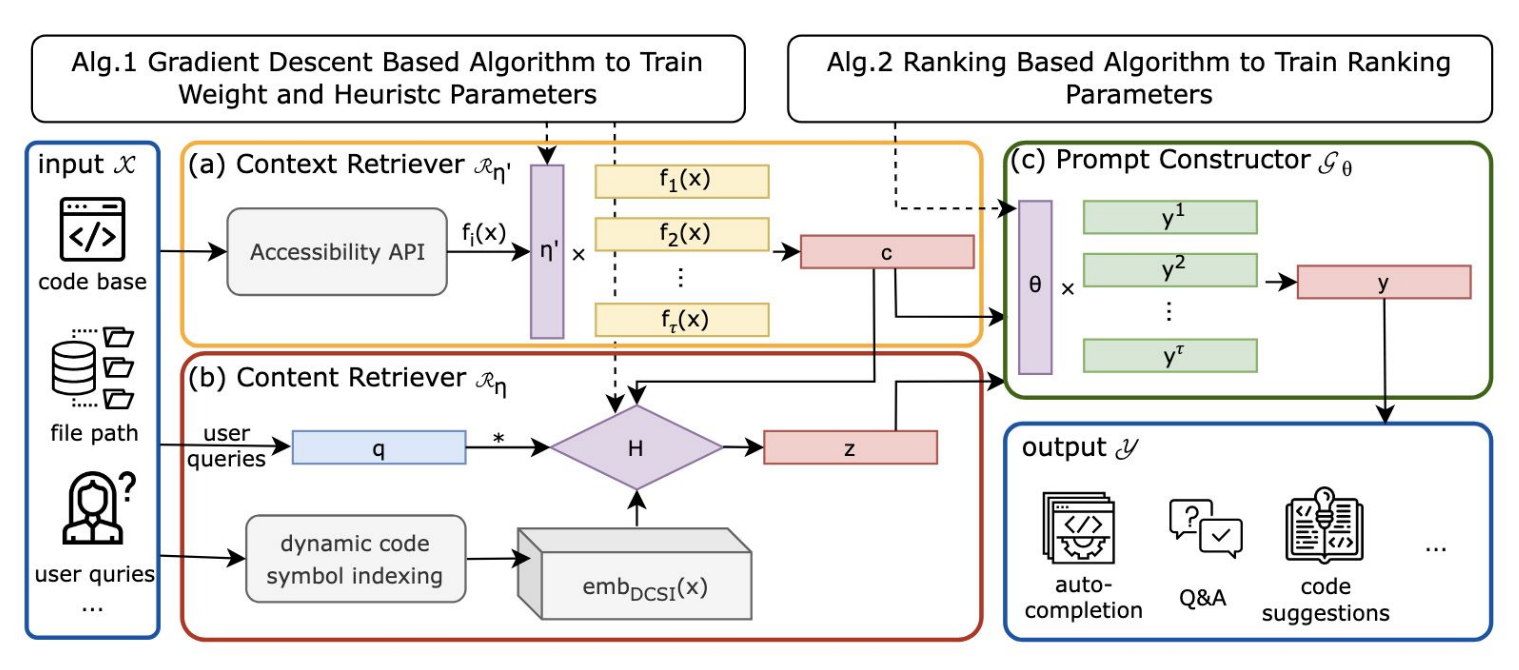
Cloud models: strong generative power, lack local context

Overview of CAMP

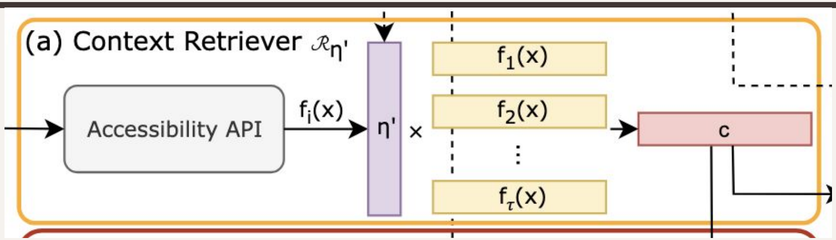
CAMP integrates cloud LLMs into local development environments, employing a RAG module that dynamically learns from code context to optimize prompt construction.



Architecture


Alg.1 Gradient Descent Based Algorithm to Train Weight and Heuristc Parameters
Alg.2 Ranking Based Algorithm to Train Ranking Parameters
input $\mathcal{X}$
code base
file path
user quries
...
(a) Context Retriever $\mathcal{R}_{\eta'}$
Accessibility API
$f_i(x)$
η'
×
$f_1(x)$
$f_2(x)$
⋮
$f_\tau(x)$
c
(b) Content Retriever $\mathcal{R}_\eta$
user queries
q
*
H
z
dynamic code symbol indexing
$emb_{DCSI}(x)$
(c) Prompt Constructor $\mathcal{G}_\theta$
θ
×
y^1
y^2
⋮
y^τ
y
output $\mathcal{Y}$
auto-completion
Q&A
code suggestions
...

Part 1. Context Retriever



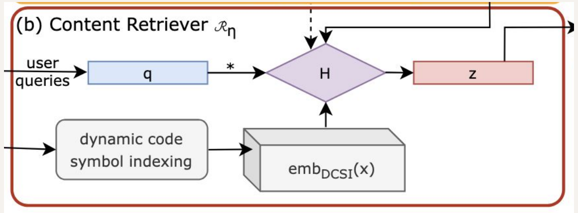
Context retriever $R_{n'}$ retrieves **contextual information** from the local development environment maximizes the insights brought to the next step.

$$\begin{aligned}\mathcal{R}_{\eta'}(x) &= \mathrm{agg}([\eta'_0 c_0, \eta'_1 c_1, \ldots, \eta'_{\tau_c} c_{\tau_c}]) \\ &= \mathrm{agg}([\eta'_0 f_0(x_0), \eta'_1 f_1(x_1), \ldots, \eta'_{\tau_c} f_{\tau_c}(x_{\tau_c})]) \\ &= \mathrm{agg}(\eta' \cdot f(x))\end{aligned}$$

Examples of c_i: cursor position, absolute repository path, cached build artifacts", index information.

Assumption: The relative importance of different factors in the local development environment remains stable.

Part 2. Content Retriever


(b) Content Retriever $\mathcal{R}_\eta$
user queries
q
*
H
z
dynamic code symbol indexing
$\text{emb}_{\text{DCSI}}(x)$

Given the retrieved context from previous step, the content retriever R_n aims to deliver highly relevant content z that enhances prompt construction with local, context-aware information.

- Function 1: support codebase embedding, with dynamic code symbol indexing.

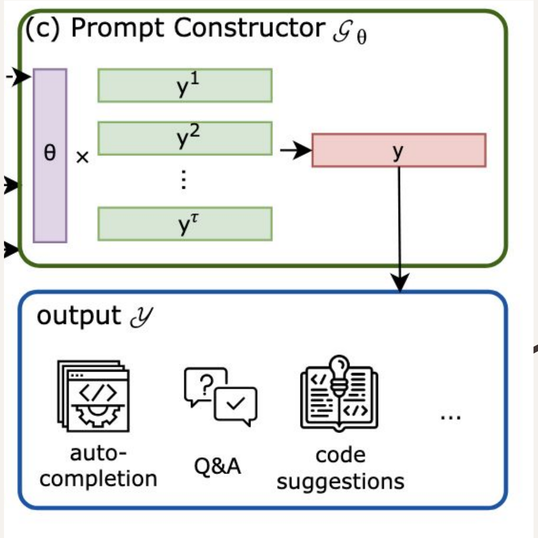
$$p_\eta(z|x,c) = \frac{\exp\left(\text{emb}_{\text{DCSI}}(z)^T H \text{emb}_{\text{DCSI}}(x)\right)}{\sum_{z'} \exp\left(\text{emb}_{\text{DCSI}}(z')^T H \text{emb}_{\text{DCSI}}(x)\right)}$$

- Function 2: facilitating content search to obtain the highest ranked content that matches the contextual information including user queries.

Part 3. Prompt Generator

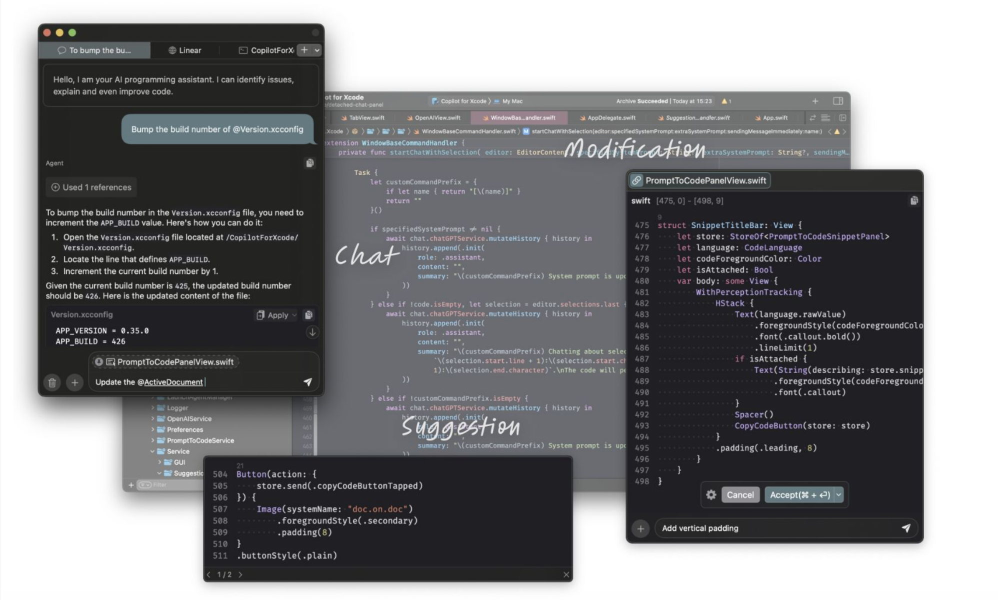
The prompt constructor G aims to determine the optimal combination and ranking of the components.

$$\begin{aligned}\mathcal{G}_\theta(x, c, z, y_{1:i-1}) = y_i &= \text{order}([y^1, y^2, \ldots, y^{\tau_k}]) \\ &= [\theta_1 \quad \theta_2 \quad \ldots \quad \theta_k]^T [y^1 \quad y^2 \quad \ldots \quad y^k]^T\end{aligned}$$



Implementation Details on Xcode

CAMP is implemented as a plugin for Xcode, to demonstrate its practical utility and validate the methodology's robustness in challenging coding environments.



Hello, I am your AI programming assistant. I can identify issues, explain and even improve code.

Bump the build number of @Version.xcconfig

To bump the build number in the `Version.xcconfig` file, you need to increment the `APP_BUILD` value. Here's how you can do it:

1. Open the `Version.xcconfig` file located at `/CopilotForXcode/Version.xcconfig`.
2. Locate the line that defines `APP_BUILD`.
3. Increment the current build number by 1.

Given the current build number is `425`, the updated build number should be `426`. Here is the updated content of the file:

Version.xcconfig

```
APP_VERSION = 0.35.0
APP_BUILD = 426
```

PromptToCodePanelView.swift

Update the @ActiveDocument

```
extension WindowBaseCommandHandler {
    private func startChatWithSelection( editor: EditorConten

        Task {
            let customCommandPrefix = {
                if let name { return "[\(name)]" }
                return ""
            }()

            if specifiedSystemPrompt ≠ nil {
                await chat.chatGPTService.mutateHistory { history in
                    history.append(.init(
                        role: .assistant,
                        content: "",
                        summary: "\(customCommandPrefix) System prompt is upc
                    ))
                }
            } else if !code.isEmpty, let selection = editor.selections.last {
                await chat.chatGPTService.mutateHistory { history in
                    history.append(.init(
                        role: .assistant,
                        content: "",
                        summary: "\(customCommandPrefix) Chatting about selec
                            `\(selection.start.line + 1):\(selection.start.ch
                            1):\(selection.end.character)`.\nThe code will pe
                    ))
                }
            } else if !customCommandPrefix.isEmpty {
                await chat.chatGPTService.mutateHistory { history in
                    history.append(.init(
                        role: .assistant,
                        content: "",
                        summary: "\(customCommandPrefix) System prompt is upc
                    ))
```

PromptToCodePanelView.swift

```
struct SnippetTitleBar: View {
    let store: StoreOf<PromptToCodeSnippetPanel>
    let language: CodeLanguage
    let codeForegroundColor: Color
    let isAttached: Bool
    var body: some View {
        WithPerceptionTracking {
            HStack {
                Text(language.rawValue)
                    .foregroundStyle(codeForegroundColo
                    .font(.callout.bold())
                    .lineLimit(1)
                if isAttached {
                    Text(String(describing: store.snipp
                        .foregroundStyle(codeForeground
                        .font(.callout)
                }
                Spacer()
                CopyCodeButton(store: store)
            }
            .padding(.leading, 8)
        }
    }
}
```

Add vertical padding

```
Button(action: {
    store.send(.copyCodeButtonTapped)
}) {
    Image(systemName: "doc.on.doc")
        .foregroundStyle(.secondary)
        .padding(8)
}
.buttonStyle(.plain)
```

Implementation Details on Xcode

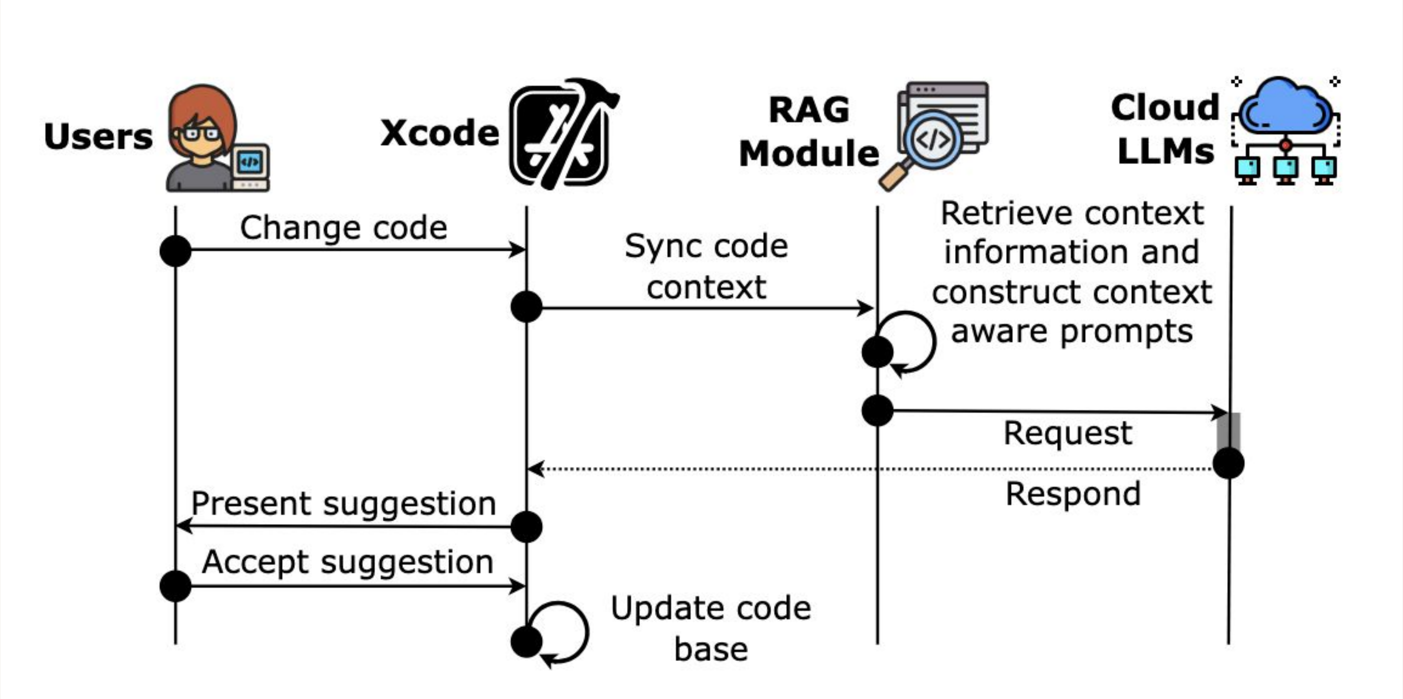
Technical challenges and solutions:

- XPC service-level communication -> enable interaction with language servers and facilitate real-time code suggestions in the UI
- Accessibility API -> capture rich contextual data

Copilot for Xcode is later assimilated to Github Copilot in October 2024.

Copilot for Xcode: exploring AI-assisted programming by prompting cloud-based large language models
CW Tan, S Guo, MF Wong, CN Hang, 2023, https://arxiv.org/abs/2307.14349
Y Wang, S Guo, CW Tan, From Code Generation to Software Testing: AI Copilot with Context-Based RAG, IEEE Software, 2025

System Flow


Users
Xcode
RAG Module
Cloud LLMs
Change code
Sync code context
Retrieve context information and construct context aware prompts
Request
Respond
Present suggestion
Accept suggestion
Update code base

Evaluation

Experiment Setup

- Benchmark: CoderEval
  - ***class-runnable***: Code outside the function but within the same class.
  - ***file-runnable***: Code outside the class but within the same file
  - ***project-runnable***: Code outside the file, spanning multiple files or repositories.

Evaluation

Baseline Models

- We compare CAMP against the following baseline models, using GPT-3.5-Turbo as the cloud-based LLM.
  - ***CloudOnly***: Inputs are processed solely by the cloud-based model, with no local processing or context retrieval.
  - ***BaseRAG***: Implements standard RAG techniques.
  - ***FileContext***: A variant of \camp{} that prioritizes context retrieved from the currently open files in the IDE. This lightweight version balances performance and resource efficiency.

Evaluation

CAMP consistently outperforms baselines

- +12.5% vs. CloudOnly
- +6.3% vs. BaseRAG
- Best results for complex, multi-file tasks

Analysis and discussion

- CAMP's context-aware code content retrieval enhances prompts
- FileContext is competitive at simple tasks, but falls short in large-scale ones

Insights and learnings

- Broader context contributes better to complex tasks.
- Dynamic scope tuning saves compute on simple tasks without sacrificing accuracy.

Future Directions

We propose directions for future research works, in algorithm optimization, model refinement, and software features extensions.

- **End to End Training**: To train all parameters end-to-end in one data pipeline, which might bring new insights to our understanding of the model parameters.
- **Trust AI**: To implement a robust user consent mechanism that empowers users with the ability to control the data access and make informed decisions about the extent of information shared.
- **Software Feature Extensions**: To develop a portable functionality kit to maximize the tool's compatibility with future versions of Xcode and multiple programming languages.