FACET: Fox-guided AI for Contextual Editing and Tagging in Interactive Image Annotation and Dataset Management

Technical Documentation Team
YipYap Project
🦊

July 11, 2025

Abstract

We introduce FACET (Fox-guided AI for Contextual Editing and Tagging), a novel AI assistant deeply integrated into the YipYap image annotation platform. FACET leverages local Large Language Models (LLMs) powered by Ollama, providing real-time, context-aware assistance for tasks such as image dataset organization, tagging, captioning, and Git-based version control. The system features a robust tool-calling mechanism, enabling the LLM to interact directly with the underlying YipYap data and Git managers. This paper details FACET's architecture, core functionalities, integration methodologies, and performance considerations, highlighting its role in enhancing user productivity and streamlining complex dataset management workflows.

1 Introduction

Modern image annotation and dataset management often involve complex, repetitive tasks that can be time-consuming and error-prone. The integration of AI assistants, particularly those powered by Large Language Models (LLMs), presents a significant opportunity to streamline these workflows. FACET addresses these challenges by providing an intelligent, context-aware assistant directly within the YipYap application, allowing users to interact naturally and efficiently with their datasets.

2 System Architecture

2.1 Core Components

FACET consists of several interconnected components, designed for seamless integration and optimal performance:

1. Ollama Client Integration
2. YipYap Assistant Core
3. Tool Calling System
4. Context Management
5. Streaming Interface

2.2 Architectural Flow

The FACET system operates through a tightly integrated frontend-backend architecture. On the backend, the `OllamaManager` initializes and manages the `OllamaClient` and the `YipYapAssistant` instance. This setup occurs during the application's lifespan, ensuring the assistant is ready upon startup. The `YipYapAssistant` itself is responsible for building the system prompt, managing conversation history, and orchestrating tool calls.

On the frontend, a SolidJS composable `useOllama` interacts with the backend `/api/ollama` endpoints. The `YipYapAssistant` UI component provides the chat interface, handling user input, displaying streaming responses, and visualizing tool execution.

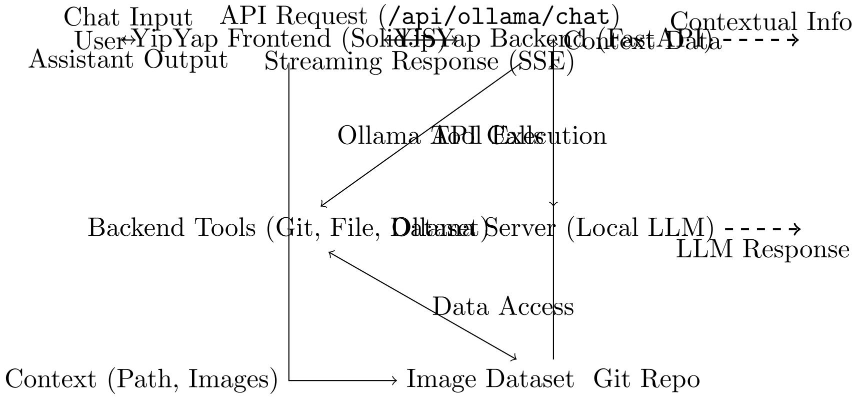

Figure 1: FACET System Architecture and Data Flow

3 Algorithmic Implementation

3.1 Context-Aware Prompt Engineering

The `YipYapAssistant` constructs a dynamic system prompt to provide the LLM with relevant context. This includes:

- A predefined personality (e.g., , friendly, knowledgeable about dataset management).
- Details about YipYap's features and functionalities.
- A dynamically generated list of available tools, including their names, descriptions, and parameters, ensuring the LLM understands its capabilities.
- Relevant memories retrieved from a `MemoryManager`, providing historical context from previous interactions.

The system also injects real-time contextual information into user messages, such as the current directory path, selected images, and Git repository status, enabling the LLM to provide highly relevant assistance.

3.2 Tool Calling Mechanism

A pivotal feature of FACET is its robust tool-calling system. The `YipYapAssistant` can parse tool call requests from the LLM's responses and execute them through a `ToolRegistry`. This allows the LLM to perform actions like:

- Git operations (e.g., `git status`, `git commit`).
- File operations (e.g., directory listing, file information).
- Dataset management (e.g., organization, statistics, analysis).

The tool execution is handled asynchronously, with results streamed back to the LLM and the user, facilitating complex multi-step interactions.

3.3 Streaming and Responsiveness

FACET utilizes Server-Sent Events (SSE) for real-time streaming of LLM responses to the frontend. This provides immediate feedback to the user, enhancing the interactive experience. The frontend's SolidJS composable `useOllama` manages the streaming state, updating the UI as new chunks of response or "thinking" content arrive. An `AbortController` is used to allow users to stop ongoing generation if needed.

4 Performance Considerations

FACET's performance is heavily influenced by the underlying Ollama setup and the chosen LLM. Key optimization strategies include:

- **Local LLM Inference**: By leveraging Ollama for local model execution, FACET minimizes network latency and ensures data privacy, as all processing occurs on the user's machine.

- **Model Selection**: Users can configure the default LLM model, with recommendations provided for various use cases (e.g., `qwen3:8b` for general queries). Smaller models generally offer faster response times.

- **Hardware Acceleration**: Ollama automatically utilizes available GPU resources, significantly accelerating inference for compatible models.

- **Memory Management**: Efficient management of LLM models in memory is crucial, with recommendations to only keep necessary models pulled to optimize RAM usage.

While no specific performance benchmarks for FACET's LLM interaction were conducted in isolation, the system aims to maintain the high responsiveness characteristic of the YipYap platform, as demonstrated by the sub-3ms response times of the NEXUS collision detection system.

5 Security and Privacy

FACET is designed with a strong emphasis on security and privacy:

- **Local Processing**: All LLM inference and data processing occurs locally on the user's machine, ensuring no sensitive data leaves the user's environment.

- **No API Keys**: The system does not rely on external LLM APIs, eliminating the need for API keys and external service dependencies.

- **Data Privacy**: Conversations are not logged or transmitted externally, safeguarding user interactions.

- **Access Control**: FACET integrates with YipYap's existing authentication system, ensuring that tool execution and data access are governed by defined user roles and permissions.

6 Conclusion

FACET represents a significant advancement in interactive image annotation and dataset management by seamlessly integrating a context-aware LLM assistant directly into the YipYap platform. By combining local LLM inference with a robust tool-calling mechanism and intelligent context management, FACET empowers users with real-time, intelligent assistance for a wide array of tasks. This novel approach enhances productivity, streamlines complex workflows, and maintains a strong focus on user privacy and security. Future enhancements include multi-modal support and deeper integration with existing YipYap features.

References

[1] Ollama: Get up and running with large language models locally. https://ollama.ai/

[2] FastAPI: The Web framework for APIs. https://fastapi.tiangolo.com/

[3] SolidJS: Simple and Performant Reactive Library. https://www.solidjs.com/

[4] Technical Documentation Team. NEXUS: A High-Performance Collision Detection System for Interactive Image Annotation. YipYap Project.