Code Agent – Core Components

Sofia Billger Bergström

Table of contents

1 Code Agent – Core Components

This document explains the main building blocks of the Code Agent.

1.1 1. Language Model & Embeddings

The agent is designed to be backend-agnostic, but the default configuration uses Ollama for local LLM inference.

```
from code_agent.main import load_config, create_llm
from langchain_community.embeddings import GPT4AllEmbeddings

# Load configuration from llm_config.json
cfg = load_config()

# Create the LLM instance (defaults to ChatOllama)
llm = create_llm(cfg)

# Use GPT4All for local, CPU-based embeddings
embeddings = GPT4AllEmbeddings(client=None)
```

- `ChatOllama` is the default LLM wrapper, providing the connection to a running Ollama server.
- `GPT4AllEmbeddings` supplies a local, CPU-based vectorizer for context retrieval.

1.2 2. Persistent Vector Store

Conversational memory is managed by a persistent vector store, allowing the agent to recall past interactions.

```
from langchain_chroma import Chroma

memory_dir = ".code_agent_memory"
vectorstore = Chroma(
    collection_name="code_agent_conversations",
    embedding_function=embeddings,
    persist_directory=memory_dir,
)
```

- The store is backed by `langchain_chroma.Chroma`, a lightweight and fast vector store.
- Persistence is handled automatically within the main application loop.

1.3 3. LangGraph Flow

The agent's logic is orchestrated by a `StateGraph` from LangGraph. This graph manages the flow of the conversation:

1. **User input** (`HumanMessage`) is added to the state.
2. The **LLM** is called with the current conversation history.
3. The LLM decides if a **Tool** should be used.
4. If so, the selected **Tool is executed**, and its output is added to the state.
5. The loop continues until the LLM generates a final **Response** (`AIMessage`) for the user.

The graph is compiled into a `Runnable` that can be easily invoked.

```
from code_agent.graph import build_graph

# The graph is constructed with the LLM and the available tools
agent_runnable = build_graph(llm, tools)
```

1.4 4. Toolset

The agent's capabilities are defined by its set of tools. Each tool is a `langchain_core.tools.BaseTool`.

Tool	Description
`new-file`	Creates a new file in the repository.
`edit-file`	Modifies an existing file.
`read-file`	Reads the content of a file.
`generate-test`	Generates a pytest test file for a Python module.
`format-code`	Formats code using standard formatters.
`search-explain`	Searches for code and explains it.
`notebook`	Executes Python code in a notebook-like environment.
`general-chat`	Handles general conversation and questions.
`r-script`	Executes R code.
`linker`	Provides context-aware file and symbol linking.

Tools are created and injected into the graph at construction time. See `TOOLS.qmd` for more details.