6 Branches 9 Tags Go to file Go to file About

zvictor and github-actions[bot] chore: update TS version and publ…

b71c9ee · 2 months ago

.github	Publish to npm using …	2 months ago
cookbook	update dependencies	2 months ago
docs	replace `Brainy Flow` …	2 months ago
python	chore: update PY ver…	2 months ago
typescript	chore: update TS ver…	2 months ago
.clinerules	add new python tests	7 months ago
.envrc	new env setup	2 months ago
.gitbook.yml	add block hints	8 months ago
.gitignore	minor improvements	6 months ago
.npmrc	split package.json int…	8 months ago
.prettierignore	fix formatting	8 months ago
LICENSE	new license (MPL-2.0)	6 months ago
README.md	replace `Brainy Flow` …	2 months ago
package.json	update dependencies	6 months ago
pnpm-workspa…	fix #21	6 months ago
prettier.config.…	cleanup implementation	6 months ago
reset.d.ts	add dev tools	8 months ago
shell.nix	new env setup	2 months ago
tsconfig.json	split package.json int…	8 months ago

Caskada: More AI with less coding 🎤🐤

skada.ai/caskada

#flow #automation #ai #llm

Readme
MPL-2.0 license
Activity
Custom properties
43 stars
2 watching
12 forks
Report repository

Releases 6

TypeScript Package v2.2.0 Latest
on Sep 7

+ 5 releases

Sponsor this project

zvictor Victor Duarte

Sponsor

Learn more about GitHub Sponsors

Packages

No packages published

Contributors 6






README MPL-2.0 license

caskada

Python 64.4%
Jupyter Notebook 17.6%
TypeScript 13.1%
HTML 4.1%
CSS 0.5%
Shell 0.2%
Other 0.1%

A radically minimalist AI framework (just 300 lines! 🤯)
Build Powerful AI Agents with Minimal Code, Maximum Freedom.

Let Agents build Agents with Zero bloat, dependencies, or vendor lock-in 😮

Caskada is a framework enabling *Agentic Coding* through powerful abstractions.

It provides a simple interface for building complex AI applications based on *nested directed graphs* with shared state. It enables both humans and AI assistants to collaborate effectively on designing and implementing AI systems.

Features

- **Brain-Easy** 🧠: Intuitive for both humans and AI assistants
- **Minimalist Design** ✨: Core abstractions in just (*you heard it right!*) 300 lines of code
- **Freedom** 🔓: Zero bloat, dependencies, or vendor lock-in
- **Composable** 🧩: Build complex systems from simple, reusable components
- **Powerful** 🦾: Supports everything you love—(Multi-)Agents, Workflow, RAG, and more
- **Agentic-Coding** 🤖: Designed for AI-assisted development
- **Universal** 🌈: Works with any LLM provider or API
- **Polyglot** 🌍: Python and Typescript are both supported

Documentation

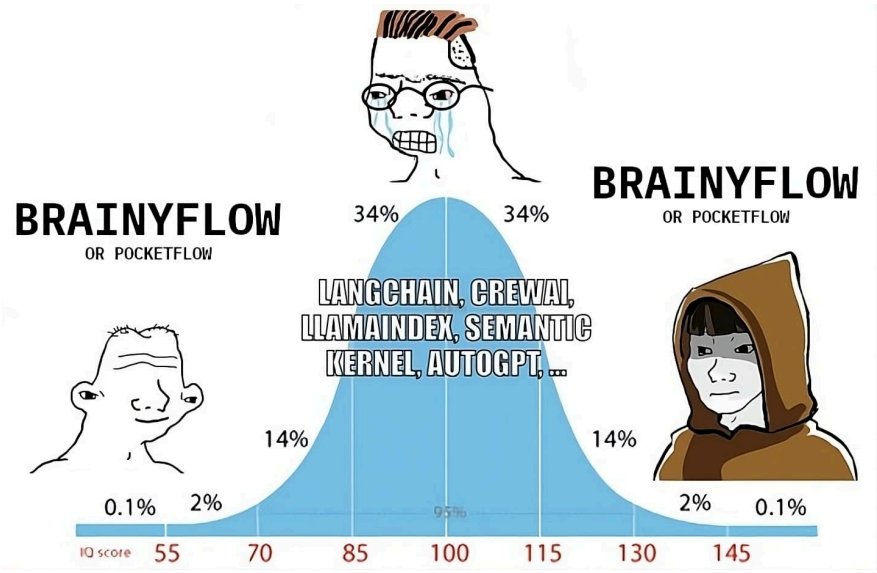
Our documentation is inclusive, suitable for both biological and synthetic minds.
Start by selecting your condition - or perhaps *the one you've been conditioned to believe*:

>> I'm Carbon-Based 🐥 <<

>> I'm Silicon-Based 🤖 <<

Why Caskada?

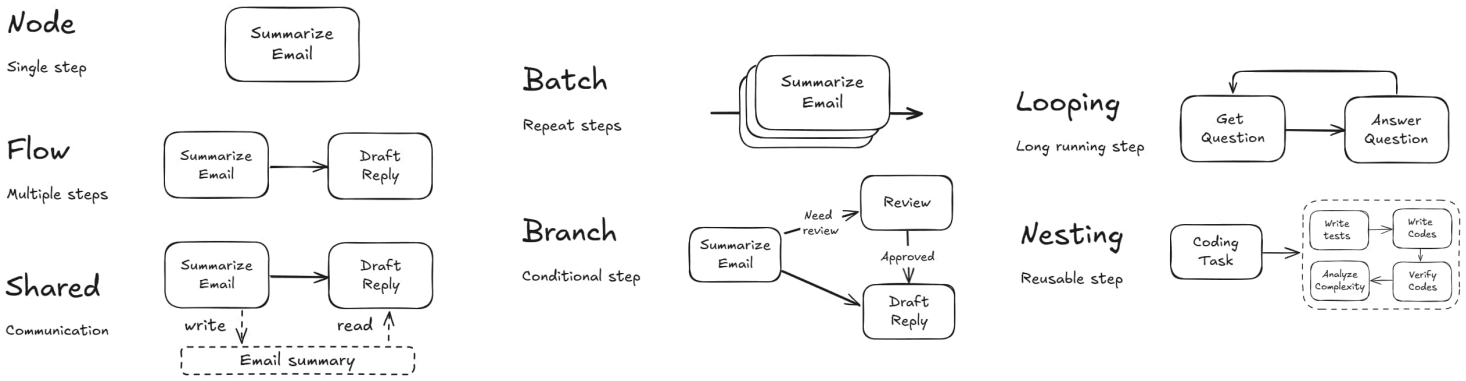
Current LLM frameworks are bloated... You actually only need 300 lines for a robust LLM Framework!



	Abstraction	App-Specific Wrappers	Vendor-Specific Wrappers	Lines	Size
LangChain	Agent, Chain	Many (e.g., QA, Summarization)	Many (e.g., OpenAI, Pinecone, etc.)	405K	+166MB
CrewAI	Agent, Chain	Many (e.g., FileReadTool, SerperDevTool)	Many (e.g., OpenAI, Anthropic, Pinecone, etc.)	18K	+173MB
SmolAgent	Agent	Some (e.g., CodeAgent, VisitWebTool)	Some (e.g., DuckDuckGo, Hugging Face, etc.)	8K	+198MB
LangGraph	Agent, Graph	Some (e.g., Semantic Search)	Some (e.g., PostgresStore, SqliteSaver, etc.)	37K	+51MB
AutoGen	Agent	Some (e.g., Tool Agent, Chat Agent)	Many [Optional] (e.g., OpenAI, Pinecone, etc.)	7K (core-only)	+26MB (core-only)
Caskada TS.ts	**Graph**	**None**	**None**	**300**	**few KB**
Caskada .py	**Graph**	**None**	**None**	**300**	**few KB**

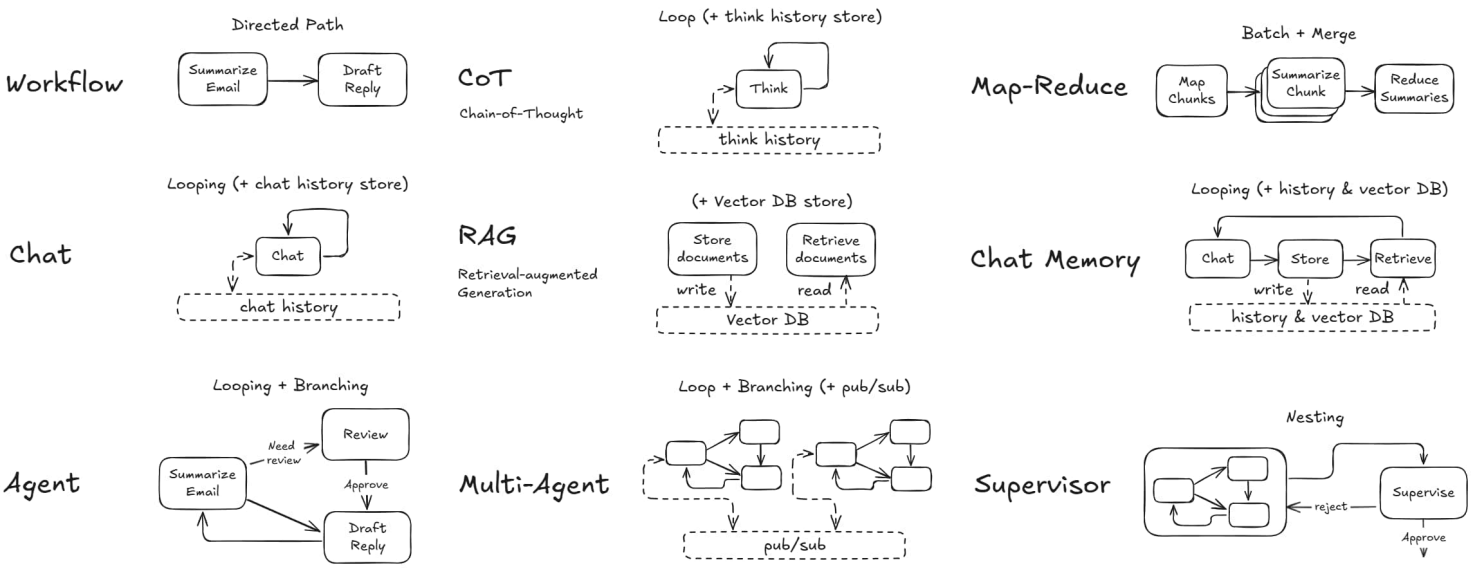
How does Caskada work?

The single file in Python or Typescript capture the core abstraction of LLM frameworks: Graph!



- Node handles simple (LLM) tasks with a clear lifecycle (`prep` → `exec` → `post`).
- Flow connects nodes through **Actions** (labeled edges), orchestrating execution.
- Memory manages shared (`global`) and isolated (`local`) state, enabling communication between nodes.

From there, it's easy to implement all popular design patterns:



- Agent autonomously makes decisions based on context.
- Workflow chains multiple tasks into sequential pipelines.
- RAG integrates data retrieval with generation.
- Map Reduce splits data tasks into Map and Reduce steps.
- Structured Output formats outputs consistently.
- Multi-Agents coordinate multiple agents.

Tutorials

Name	Difficulty	Description
Chat	☆☆☆ *Dummy*	A basic chat bot with conversation history

Name	Difficulty	Description
RAG	☆☆☆ *Dummy*	A simple Retrieval-augmented Generation process
Workflow	☆☆☆ *Dummy*	A writing workflow that outlines, writes content, and applies styling
Map-Reduce	☆☆☆ *Dummy*	A resume qualification processor using map-reduce pattern for batch evaluation
Agent	☆☆☆ *Dummy*	A research agent that can search the web and answer questions
Streaming	☆☆☆ *Dummy*	A real-time LLM streaming demo with user interrupt capability
Multi-Agent	★☆☆ *Beginner*	A Taboo word game for asynchronous communication between two agents
Supervisor	★☆☆ *Beginner*	Research agent is getting unreliable... Let's build a supervision process
Parallel	★☆☆ *Beginner*	A parallel execution demo that shows 3x speedup
Thinking	★☆☆ *Beginner*	Solve complex reasoning problems through Chain-of-Thought
Memory	★☆☆ *Beginner*	A chat bot with short-term and long-term memory

And many more available for all levels! Check them all out!

Quick Start

New to Caskada? Check out our Getting Started guide to build your first flow in no time.

Ready to Build Self-Coding Apps?

Check out Agentic Coding Guidance, the fastest way to develop self-coding LLM projects with Caskada!

Acknowledgement

We would like to extend our deepest gratitude to the creators and contributors of the PocketFlow framework, from which Caskada originated as a fork.

Contributors Wanted!

We're looking for contributors for all aspects of the project. Whether you're interested in documentation, testing, or implementing features, we'd love your help!

Get involved by joining our Discord server.

Liability Disclaimer

Caskada is provided "as is" without any warranties or guarantees.
We do not take responsibility for how the generated output is used, including but not limited to its accuracy, legality, or