



Infrastructure for LLM application ecosystems

Michael Yuan

<https://github.com/WasmEdge/WasmEdge>



SECOND STATE

LLMs are not AGI, yet

Otherwise, there will be no “applications”



LLM is a brain with anterograde amnesia

Memory & knowledge

Eyes and ears

Hands & actions

Needs to repeat the entire conversation history at every turn

Limited by the context length

No long term memory either

Cannot receive trigger events on its own.

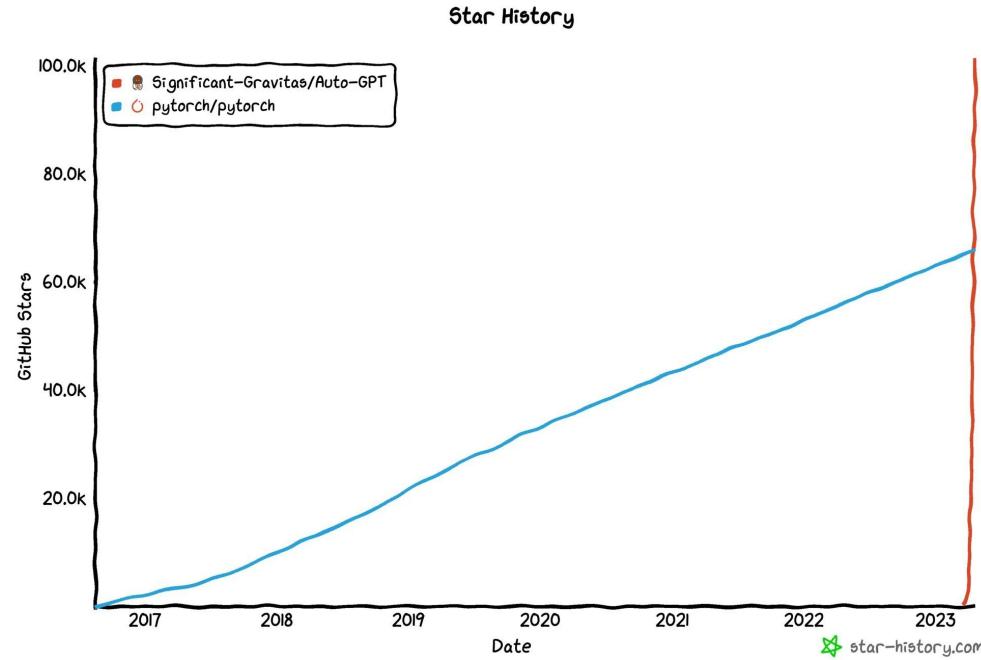
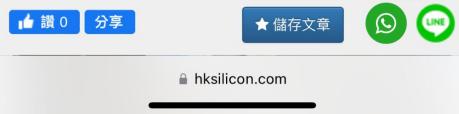
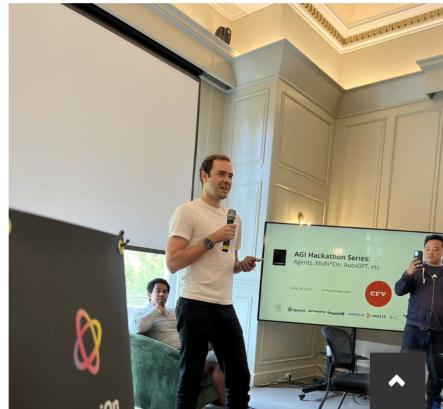
Cannot learn new knowledge.

The outputs are words only.



另一方面，Andrej Karpathy認為普通人、創業者和極客在構建AI Agents方面相比OpenAI這樣的公司更有優勢，大家目前處於平等競爭的狀態，因此他很期待看到這方面的成果。

他還透露，如果一篇論文提出了某種不同的訓練方法，OpenAI內部的Slack上會嗤之以鼻，認為都是他們玩剩下的。但是當新的AI Agents論文出來的時候，他們會認真興奮的討論。



AutoGPT, BabyAGI, AgentGPT etc.

	NATIVE APPS	COMPETITORS	FEATURING	Revenue Last 30 days	Downloads Last 30 days	Cumulative RpD Lifetime	RpD Tier-1 West Lifetime	RpD Tier-1 East Lifetime	Overlap Score
   Genie - AI Chatbot AppNation Ltd.	\$1,090,228	1,907,938	\$0.53	\$0.69	\$0.35				
Competitors									
  Chat AI - AI Chatbot Assistant SEA MOBILE LTD	\$84,155	3,107,096	< \$0.05	\$0.05	\$0.08				100%
  AI Chat: AI Smith Open Chatbot Vulcan Labs Company Limited	\$1,144,426	2,678,273	\$0.38	\$0.51	\$0.33				78%
  ChatOn - AI Chat Bot Assistant AIBY	\$1,503,173	1,281,618	\$0.76	\$0.95	\$0.42				76%
  AI Chat - Chatbot AI Assistant Social Media Apps & Games GmbH	\$832,078	1,793,883	\$0.51	\$0.73	\$0.18				76%
  Chat AI Bot: Chatbot Assistant Tresor Tech	≈ \$27,048	2,024,517	< \$0.05	—	—				71%
  Wonder - AI Art Generator Codeway Dijital Hizmetler Anonim Sirketi	\$962,025	1,007,246	\$0.44	\$0.74	\$0.1				70%
  AI Chat - Assistant & Chatbot tappz	\$218,398	726,200	\$0.22	\$0.3	\$0.13				64%
  Impulse - Brain Training GMRD Apps Limited	\$1,925,181	1,513,435	\$1.01	\$1.62	\$0.56				64%
  Roboco - AI Chatbot Assistant WEBTECH INTERNATIONAL L.P.	\$159,514	610,747	\$0.25	\$0.27	\$0.14				61%
  Chai - Chat with AI Friends Chai Research Corp.	\$285,112	985,344	\$0.45	\$0.63	\$0.59				60%
  WOMBO Dream - AI Art Generator Wombo Studios Inc	\$118,807	1,003,371	< \$0.05	\$0.11	< \$0.05				58%



Solutions

Memory & knowledge

Large context length
(16k for GPT35, 32k
for GPT4, 100k for
Claude)

Use vector database
to store and search
embeddings

Eyes and ears

Connect to LLM APIs
to Internet events.

Support outbound
web browsing and
fetching.

Hands & actions

Generate structured
data outputs.

Connect LLM
outputs to other
SaaS to perform
actions.



LLM functions and plugins

- LLM lockin
- Model workflow lockin
- UI lockin
- User must initiate the conversation
- Does not support machine encoded input or triggers

A better solution: open source frameworks

Announcing our \$10M seed round led by Benchmark



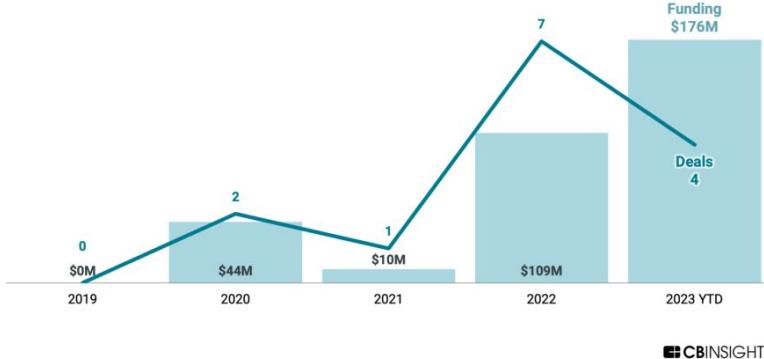
LangChain

Apr 4, 2023 • 4 min read



Funding to vector database startups takes off

Disclosed equity funding & deals (as of 04/27/2023)



CB INSIGHTS

Exclusive: AI startup LangChain taps Sequoia to lead funding round at a valuation of at least \$200 million

Stephanie Palazzolo Apr 13, 2023, 3:41 PM CDT



- Must build and manage infra – not serverless
 - Connectors and drivers for everything even for locally hosted databases etc.
 - Manage and scale resources
 - Manage auth for all external services
- Python-based solutions



Python?



Following

Greg Brockman

@gdb

President & Co-Founder [@OpenAI](#)

[gregbrockman.com](#)

Joined July 2010

67 Following 370.4K Followers



Greg Brockman

@gdb

...

Much of modern ML engineering is making
Python not be your bottleneck.

6:55 AM · 7/6/23 from Earth · 244K Views

As an analogy, model weights are like long-term memory. When you have to take a model, it's like studying for an exam a week away. When the exam arrives, the model may forget details, or struggle to recall concepts it never read.



Follow

Chris Albon

@chrisalbon

Director of ML @Wikimedia. Studies AI @ [ChrisAlbon.com](#). Created [MachineLearningFlashcards.com](#). Co-founded [@therealyonder](#).

Science & Technology ⓘ San Francisco

Joined December 2007

970 Following 77.8K Followers



Santiago Víquez ✅

@santiviquez

...

The best minds of my generation are thinking about how to install Python.



Chris Albon @chrisalbon · 1d

What is "the right way" to install Python on a new M2 MacBook? I assume it isn't the system Python3 right? Maybe Homebrew?

3:42 AM · 7/6/23 from Earth · 744K Views

Rust!



Bojan Tunguz  @tunguz · Apr 21

AGI will be built with Python.

...

Let that sink in.



519



382



5,084



3.8M



Elon Musk  

@elonmusk

...

Rust

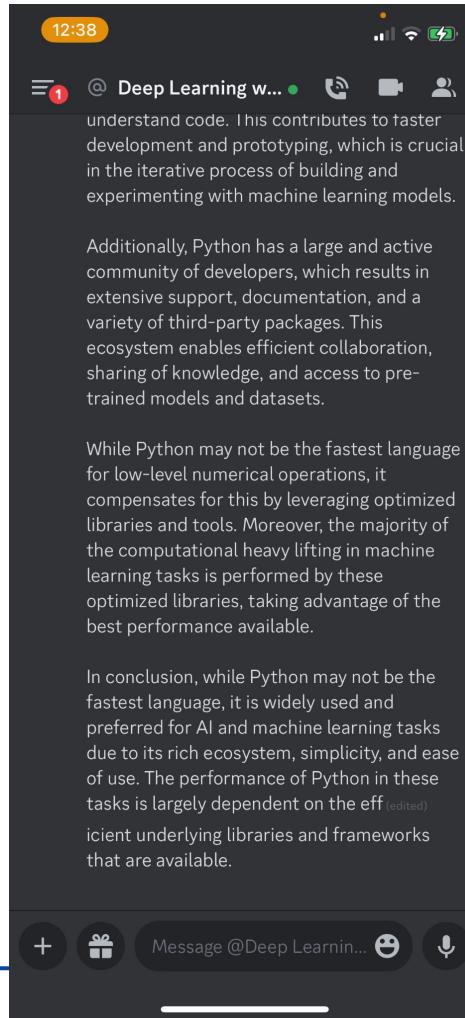
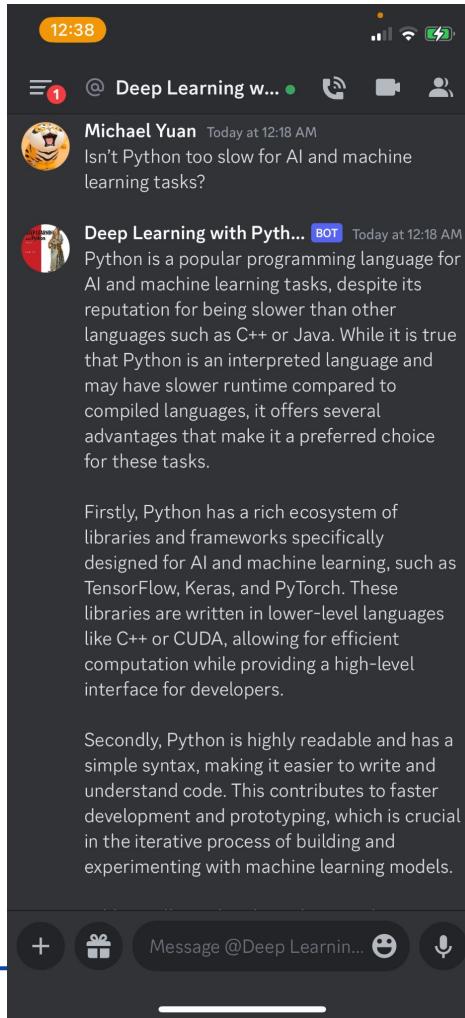
4:40 AM · Apr 22, 2023 · **3.7M** Views

682 Retweets

333 Quotes

10.4K Likes

334 Bookmarks





The Wasm approach

- Wasm is a developer friendly compiler target
 - Alternative to Python
 - Supports multiple languages (including Rust, JS and Python)
 - W3C standard with mature toolchain
- Secure sandbox for 3rd party / user contributed code
- Submillisecond startup is critical for event driven functions
- Small footprint is critical for long running networking tasks
- Supports highly efficient AI inference using Rust

X CNCF Annual Survey 2022 | ⏪ ...

 CLOUD NATIVE COMPUTING FOUNDATION 

KEY FINDINGS 3/3



CONTAINERS ARE THE NEW NORMAL, AND WEBASSEMBLY IS THE FUTURE

X CNCF Annual Survey 2022 | ⏪ ...

 CLOUD NATIVE COMPUTING FOUNDATION 

END USER ORGANIZATIONS THAT HAVE USED WEBASSEMBLY, WITH WASMEDGE AND WAMR BEING THE TOP RUNTIMES

37%

X CNCF Annual Survey 2022 | ⏪ ...

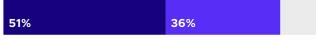
 CLOUD NATIVE COMPUTING FOUNDATION 

Have you used or are aware of the following WebAssembly runtimes?

WASMEDGE



WEBASSEMBLY MICRO RUNTIME (WAMR)



WASMER



● Yes, using or have used ● Yes, aware of the runtime ● No, not aware



WebAssembly Runtime



WasmEdgeRuntime

A lightweight, secure, high-performance and extensible WebAssembly Runtime

1. Support networking socket and web services
2. Support databases, caches, and DOs
3. Support AI inference in Tensorflow, OpenVino, PyTorch etc.
4. Seamlessly integrates into the existing cloud-native infra
5. Support writing wasm programs using JS



CLOUD NATIVE
COMPUTING FOUNDATION

<https://github.com/WasmEdge/WasmEdge>



Opinionated runtime optimized for cloud native

- 1/100 the size of typical LXC images
- 1000x faster startup time
- Near native runtime performance
- Secure by default and very small attack surface
- Completely portable across platforms
- Programming language agnostic
- Plays well with k8s, service mesh, distributed runtimes etc.

Wasm support for AI inference

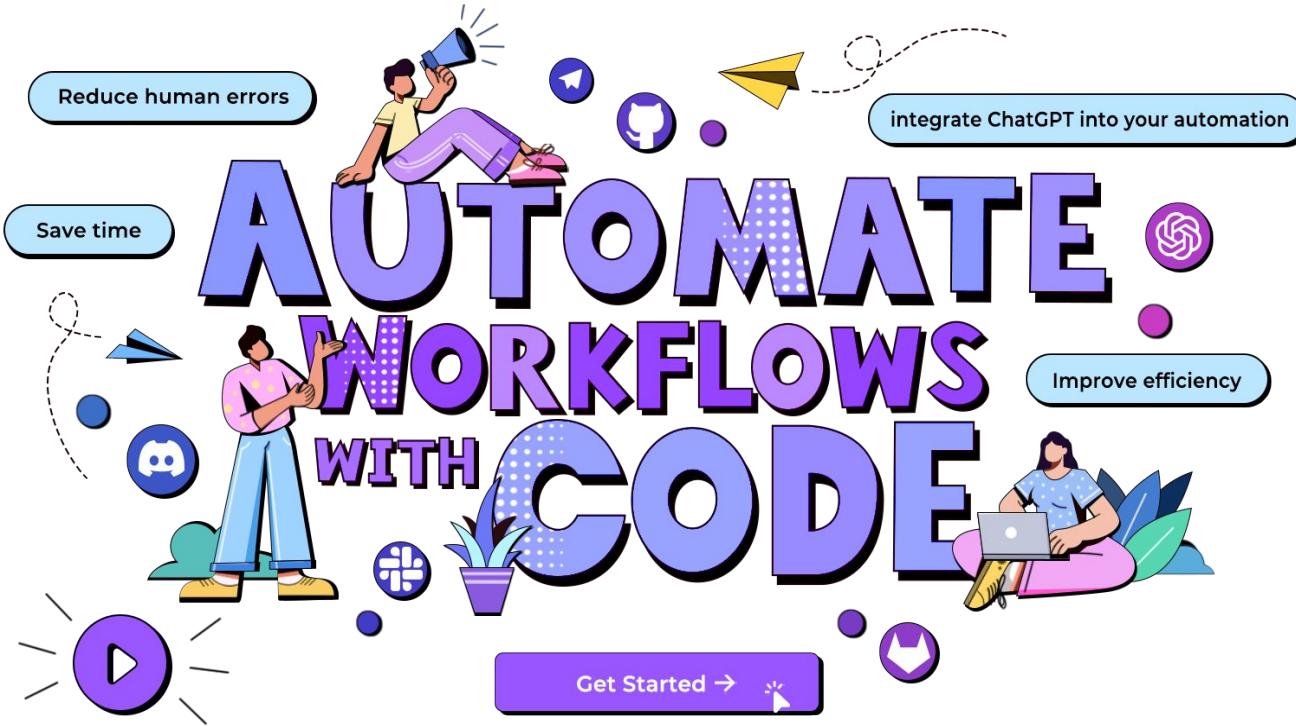
A Rust library for MediaPipe tasks for WasmEdge WASI-NN

Introduction

- **Easy to use:** low-code APIs such as mediapipe-python.
- **Low overhead:** No unnecessary data copy, allocation, and free during the processing.
- **Flexible:** Users can use custom media bytes as input.
- For TFLite models, the library not only supports all models downloaded from [MediaPipe Solutions](#) but also supports [TF Hub](#) models and [custom models](#) with essential information.

Status

- Object Detection
- Image Classification
- Image Segmentation
- Interactive Image Segmentation
- Gesture Recognition
- Hand Landmark Detection
- Image Embedding
- Face Detection
- Face Landmark Detection
- Pose Landmark Detection
- Audio Classification
- Text Classification
- Text Embedding
- Language Detection



<https://github.com/flows-network/>

Code review



Use WasmEdge + GPT4 to review your PRs!

[Rust] Optimize host function #2394
apekuss merged 8 commits into `WasmEdge:master` from `apekuss:rust/refactor-function` on Apr 2

Merged

alabulei1 commented on Apr 5

Hello, I am a [serverless review bot](#) on [flows.network](#). Here are my reviews of code commits in this PR.

Overall Summary:

This set of patches involves modifications primarily to the Rust bindings for the wasmedge-sys component, the related CI workflow, and the test files. The primary focus of these changes is to improve memory management, remove limits on the number of host functions, and update assertions in tests.

Potential issues and errors:

1. Removing the check for host function capacity and the 'MAX_HOST_FUNC_LENGTH' environment variable could lead to excessive resource consumption, affecting system stability and performance.
2. The `test_crazy_function_creation` function may consume a significant amount of system resources and take a long time to finish. Additionally, increasing the loop count from 10 to 1,000,000 in `function.rs` might lead to performance concerns and resource exhaustion.
3. Changing the assertions in tests to use `assert!` instead of `assert_eq!` (and vice versa) could lead to altered test outcomes and potential false test failures.
4. Mutexes and RwLocks in the code can potentially cause deadlocks if not implemented carefully.
5. Updating the Rust CI workflow to restrict the number of test threads to 1 across all platforms could slow down the CI process.

Important Findings:

1. Memory management has been improved through the addition of support for removing real functions in Rust, and the creation of a footprint for host functions.
2. Changes have been made to the Rust bindings CI workflow, including updating the operating systems matrix and Rust compiler versions matrix, as well as modifying the cargo test commands to set the test-threads to 1.
3. The various changes to the test files involve updating assertions and modifying test scenarios, affecting the conditions that are being checked and the performance of the tests.

Given these changes, it's important to ensure that potential issues such as deadlocks, resource exhaustion, and false test failures are mitigated. Also, be cautious when removing limitations on host functions, as it could impact overall system stability and performance.

[WIP] plugin opencvmini #2403
dannypsnl wants to merge 11 commits into `master` from `plugin-opencvmini`

Open

juntao commented last month

Hello, I am a [serverless review bot](#) on [flows.network](#). Here are my reviews of changed source code files in this PR.

plugins/wasmedge_opencvmini/opencvmini_base.h

The code snippet provided is a C++ code for a `WasmEdgeOpenCVMini` class. Here are some observations:

1. `#pragma once` is used to prevent multiple inclusions of the same file which is fine.
2. `SPDX-License-Identifier` and `SPDX-FileCopyrightText` are used to indicate the license.
3. The `opencvmini_env.h` header file and two other header files are included.
4. A template class named `WasmEdgeOpenCVMini` that inherits from the `Runtime::HostFunction<T>` class is defined.
5. The constructor initializes the inherited class `HostFunction<T>` with a value zero and an instance of `WasmEdgeOpenCVMiniEnvironment`.
6. The class declaration is enclosed within two namespace blocks, `WasmEdge` and `Host`.

There are no visible problems with the provided code snippet. This seems to be a class definition and more code may be added later.

The patch doesn't introduce any changes to the existing code. It is the same as the original code snippet.

plugins/wasmedge_opencvmini/opencvmini_env.cpp

This code snippet defines a plugin for the OpenCVMini library in the context of the WasmEdge project. The code sets up a `WasmEdgeOpenCVMiniEnvironment` class that derives from `Plugin::HostEnvironment`. There is a module `WasmEdgeOpenCVMiniModule` that is created and stored in the `Plugin::PluginModule::ModuleDescriptor` object `Descriptor`. This object includes the name of the plugin, its version, and other related information. The plugin descriptor contains only one module, namely `WasmEdgeOpenCVMiniModule`.

The code snippet seems to be correct and without any obvious problems. However, there may be issues with this code depending on the context in which it is used, which cannot be reviewed from this snippet alone.

The patch has no substantive changes. It is simply reformatting the code, reorganizing the newlines between some of the statements, and adjusting the indentation. Overall, it looks like style and formatting updates, and it does not introduce any functional changes to the code logic.

Add function 'check_prime' for node's crypto API by Aviii06 · Pull Request #82

Changes from all commits ▾ File filter ▾ Conversations ▾ ⚙ ▾

Filter changed files

example_js/node

- main.mjs
- package.json
- rollup.config.js

src

49 src/internal_module/crypto.rs

```
16 + if n <= 1 {  
17 +     return JsValue::Bool(false);  
18 + }  
19 + let limit = (n as f64).sqrt() as i32;  
20 + for a in 2..limit {  
21 +     if n % a == 0 {  
22 +         return JsValue::Bool(false);  
23 +     }  
24 + }  
25 + JsValue::Bool(true)  
26 + }  
27 +
```



second-state/wasmedge-quickjs Add function 'check_prime' for node's crypto API



Potential problems:

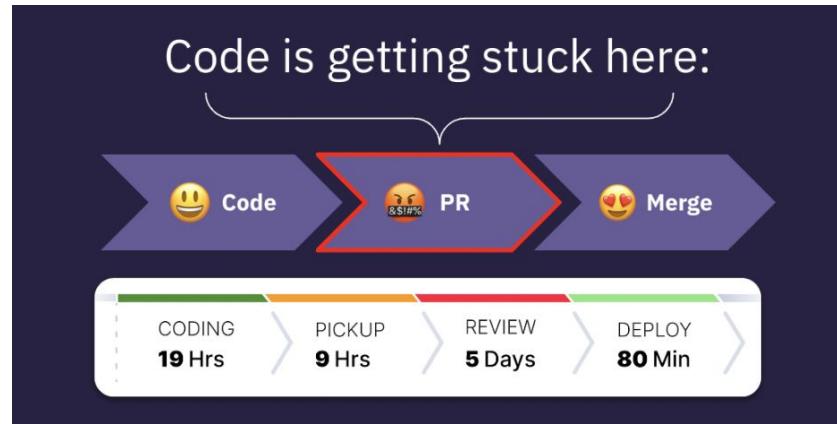
1. The check_prime function can be optimized further, as it checks for divisibility with even numbers after 2, which isn't necessary.

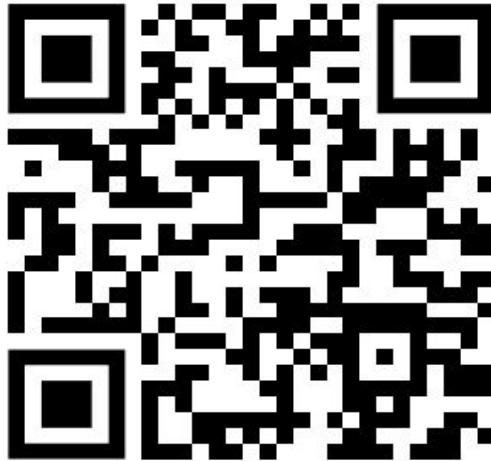


Re-discover the Sieve of Eratosthenes

- Ask follow up questions in a conversation
 - What about multiples of 3?
 - What about multiples of 5?
 - What about multiples of primes we already knew?
- Sieve of Eratosthenes
 - One of the most efficient ways to check for prime numbers under 10m
 - Discovered by Eratosthenes over 2200 years ago

- Senior developers are very busy and very expensive.
- Yet, the development process cannot move forward (eg merging the PR) without the code review. Developers are often idle waiting for reviews.





<https://github.com/flows-network/github-pr-summary>

1. **Load the code review bot template in flows.network.** The template contains the source code for the bot itself. We will clone the source code to your own GitHub account so that you can modify and customize it later. Click on Create and Deploy.
2. **Authorize bot access to GitHub.** The `github_owner` and `github_repo` point to the target GitHub repo where the bot will review PRs. Click on Authorize to give the repo the necessary permissions in GitHub.
3. **Give the bot your OpenAI API key.** If you have saved API keys in the past, you can skip this step and reuse these keys.



```
#[no_mangle]
#[tokio::main(flavor = "current_thread")]
pub async fn run() -> anyhow::Result<()> {
    dotenv().ok();
    logger::init();
    log::debug!("Running github-pr-summary/main");

    let owner = env::var("github_owner").unwrap_or("juntao".to_string());
    let repo = env::var("github_repo").unwrap_or("test".to_string());
    let trigger_phrase = env::var("trigger_phrase").unwrap_or("flows summarize".to_string());

    let events = vec!["pull_request", "issue_comment"];
    listen_to_event(&GithubLogin::Default, &owner, &repo, events, |payload| {
        handler(
            &owner,
            &repo,
            &trigger_phrase,
            payload,
        )
    })
    .await;
}

Ok(())
}
```

Step 1: The application registers with a Rust host app to receive external trigger events.

When the event is received, the host will call `run()` again and `listen_to_event()` will be able to retrieve the event data in the payload.

```
let pulls = octo.pulls(owner, repo);
let patch_as_text = pulls.get_patch(pull_number).await.unwrap();
let mut current_commit = String::new();
let mut commits: Vec<String> = Vec::new();
for line in patch_as_text.lines() {
    if line.starts_with("From ") {
        // Detected a new commit
        if !current_commit.is_empty() {
            // Store the previous commit
            commits.push(current_commit.clone());
        }
        // Start a new commit
        current_commit.clear();
    }
    // Append the line to the current commit if the current commit is
    if current_commit.len() < CHAR_SOFT_LIMIT {
        current_commit.push_str(line);
        current_commit.push('\n');
    }
    if !current_commit.is_empty() {
        // Store the last commit
        commits.push(current_commit.clone());
    }
}
```

Step 2: The handler() function uses GitHub Rust SDK to retrieve all patches associated with commits in the PR.

The PR information is passed to the function via the payload.

```
let chat_id = format!("PR#{pull_number}");
let system = &format!("You are an experienced software developer. You will act as a reviewer for
let mut openai = OpenAIFlows::new();
openai.set_retry_times(3);

let mut reviews: Vec<String> = Vec::new();
let mut reviews_text = String::new();
for (_i, commit) in commits.iter().enumerate() {
    let commit_hash = &commit[5..45];
    log::debug!("Sending patch to OpenAI: {}", commit_hash);
    let co = ChatOptions {
        model: MODEL,
        restart: true,
        system_prompt: Some(system),
    };
    let question = "The following is a GitHub patch. Please summarize the key changes and identify the main purpose of the patch.";
    match openai.chat_completion(&chat_id, &question, &co).await {
        Ok(r) => {
            if reviews_text.len() < CHAR_SOFT_LIMIT {
                reviews_text.push_str("-----\n");
                reviews_text.push_str(&r.choice);
                reviews_text.push_str("\n");
            }
            let mut review = String::new();
            review.push_str(&format!("### [Commit {commit_hash}](https://github.com/WasmEdge/WasmEdge/pull/{pull_number}/commit/{commit_hash})\n"));
            review.push_str(&r.choice);
            review.push_str("\n\n");
            reviews.push(review);
            log::debug!("Received OpenAI resp for patch: {}", commit_hash);
        }
        Err(e) => {
            log::error!("OpenAI returned an error for commit {commit_hash}: {}", e);
        }
    }
}
```

Step 3: Each patch is sent to ChatGPT for summarization. The commit patch summaries are stored in an array.

```
let mut resp = String::new();
resp.push_str("Hello, I am a [code review bot](https://github.com/flows-network/github-pr-
if reviews.len() > 1 {
    log::debug!("Sending all reviews to OpenAI for summarization");
    let co = ChatOptions {
        model: MODEL,
        restart: true,
        system_prompt: Some(system),
    };
    let question = "Here is a set of summaries for software source code patches. Each summ.
match openai.chat_completion(&chat_id, &question, &co).await {
    Ok(r) => {
        resp.push_str(&r.choice);
        resp.push_str("\n\n## Details\n\n");
        log::debug!("Received the overall summary");
    }
    Err(e) => {
        log::error!("OpenAI returned an error for the overall summary: {}", e);
    }
}
for (_i, review) in reviews.iter().enumerate() {
    resp.push_str(review);
}

// Send the entire response to GitHub PR
// issues.create_comment(pull_number, resp).await.unwrap();
match issues.update_comment(comment_id, resp).await {
    Err(error) => {
        log::error!("Error posting resp: {}", error);
    }
    _ => {}
}
```

Step 4: Use ChatGPT API to summarize the summaries and send the result back to the PR as a comment.

Learning assistant (copilot)

Rust IN ACTION

Systems programming concepts
and techniques

Tim McNamara

MANNING



1:55

1 @ Rust in Action



Michael Yuan Today at 1:55 AM

What are reasons people like Rust so much?



Rust in Action BOT Today at 1:55 AM

There are several reasons why people like Rust so much:

1. Memory safety: Rust's ownership system and borrowing system ensure that programs are memory safe by preventing common bugs such as null pointer dereferences, buffer overflows, and data races. This makes Rust programs highly reliable and secure.
2. Performance: Rust offers low-level control over memory allocation and efficient abstractions, allowing developers to write high-performance code. This makes Rust suitable for systems programming and performance-sensitive applications.
3. Concurrency: Rust provides built-in concurrency features like threads, async/await, and message passing, making it easy to write concurrent and parallel programs while ensuring thread safety.
4. Expressive and safe abstractions: Rust's expressive syntax and powerful

1:56

1 @ Rust in Action



Michael Yuan Today at 1:55 AM

Can you elaborate on #3 with an example?



Rust in Action BOT Today at 1:55 AM

Certainly! One of the key features of Rust is its ability to handle concurrency in a safe and efficient manner. The `async/await` syntax in Rust allows developers to write asynchronous code that can efficiently execute concurrent tasks without introducing data races or other concurrency-related bugs.

Here's a simple example to illustrate how Rust enables safe concurrency with `async/await`:

```
use std::time::Duration;
use tokio::time::delay_for;

async fn greet(name: &str) {
    delay_for(Duration::from_secs(1)).await;
    println!("Hello, {}!", name);
}

#[tokio::main]
async fn main() {
    let countries = vec!["USA", "France", "Japan", "Brazil", "Australia"];

    let greet_tasks =
        countries.into_iter().map(|country| greet(country));
    tokio::join!(task1, task2);
}
```

1:57

1 @ Rust in Action



Michael Yuan Today at 1:57 AM

I would like to greet countries instead of people. Can you make that change to the code?



Rust in Action BOT Today at 1:57 AM

Certainly! Here's an updated version of the code that greets countries instead of people:

```
use std::time::Duration;
use tokio::time::delay_for;

async fn greet(country: &str) {
    delay_for(Duration::from_secs(1)).await;
    println!("Hello, {}!", country);
}

#[tokio::main]
async fn main() {
    let countries = vec!["USA", "France", "Japan", "Brazil", "Australia"];

    let greet_tasks =
        countries.into_iter().map(|country| greet(country));
    tokio::join_all(greet_tasks).await;
}
```

```
async fn handler(bot: &ProvidedBot, bot_id: u64, cs: &ContentSettings, msg: Message) {
    let discord = bot.get_client();
    if msg.author.bot {
        log::debug!("ignored bot message");
        return;
    }
    if msg.member.is_some() {
        let mut mentions_me = false;
        for u in &msg.mentions {
            log::debug!("The user ID is {}", u.id.as_u64());
            if *u.id.as_u64() == bot_id {
                mentions_me = true;
                break;
            }
        }
        if !mentions_me {
            log::debug!("ignored guild message");
            return;
        }
    }

    let channel_id = msg.channel_id;
    log::info!("Received message from {}", channel_id);
    let help = format!("Ask any question about the '{}' book, by {} from {}. Get the book's summary or ask a related question.", title, author, year);

    let mut text = String::from(&msg.content);
    let mut random_q = String::new();
    if text.eq_ignore_ascii_case("/help") {
        _ = discord.send_message(
            channel_id.into(),
            &serde_json::json!({
                "content": &help
            }),
        ).await;
        return;
    }

    } else if text.eq_ignore_ascii_case("/start") {
        _ = discord.send_message(
            channel_id.into(),
            &serde_json::json!({
                "content": "Welcome to the BookBot! I'm here to answer your questions about books. Just type a question and I'll do my best to provide an accurate answer. You can also ask for a book's summary or related information. To get started, type '/help' to see what I can do."
            })
    }
}
```

Step 1: The handler() function is called when a message is received by the bot.

It determines whether it should respond to the message.

If the message is a “slash command”, the bot will respond according to predefined rules.

```
let mut question_history = String::new();
match chat_history(&channel_id.to_string(), 10) {
    Some(v) => {
        for m in v.into_iter() {
            if let ChatRole::User = m.role {
                question_history.push_str(&m.content);
            }
        }
    },
    None => (),
};

question_history.push_str(&text);
// Compute embedding for the question
let question_vector = match openai.create_embeddings(EmbeddingsInput::String(question_history))
    Ok(r) => {
        if r.len() < 1 {
            log::error!("OpenAI returned no embedding for the question");
            _ = discord.edit_message(
                channel_id.into(), placeholder.id.into(),
                &serde_json::json!({
                    "content": &cs.error_mesg
                }),
                .await;
            return;
        }
        r[0].iter().map(|n| *n as f32).collect()
    }
    Err(e) => {
        log::error!("OpenAI returned an error: {}", e);
        _ = discord.edit_message(
            channel_id.into(), placeholder.id.into(),
            &serde_json::json!({
                "content": &cs.error_mesg
            }),
            .await;
        return;
    }
};
```

Step 2: Use the user's question history to create an embedding.

```

// Search for embeddings from the question
let p = PointsSearchParams {
    vector: question_vector,
    limit: 5,
};

let mut system_prompt_updated = String::from(&cs.system_prompt);
match search_points(&cs.collection_name, &p).await {
    Ok(sp) => {
        for p in sp.iter() {
            if system_prompt_updated.len() > SOFT_CHAR_LIMIT { break; }
            system_prompt_updated.push_str("\n");
            system_prompt_updated.push_str(p.payload.as_ref().unwrap().get("text").unwrap());
            // log::debug!("Adding prompt {}", first_x_chars(p.payload.as_ref().unwrap().get(
        }
    }
    Err(e) => {
        log::error!("Vector search returns error: {}", e);
        _ = discord.edit_message(
            channel_id.into(), placeholder.id.into(),
            &serde_json::json!({
                "content": &cs.error_mesg
            }),
        ).await;
        return;
    }
}

```

Step 3: Search the book contents' embeddings using the questions.

Put the returned book content into the context of the conversation.

Note: We have broken down the book into logical chunks and created an embedding for each chunk.

Sections from the book that are related to the questions will be returned by the search.

```
let mut co = ChatOptions::default();
// co.model = ChatModel::GPT4;
co.model = ChatModel::GPT35Turbo16K;
co.restart = restart;
co.system_prompt = Some(&system_prompt_updated);
// co.max_tokens = Some(500);

match openai.chat_completion(&channel_id.to_string(), &text, &co).await {
    Ok(r) => {
        let resp = sub_strings(&r.choice, 1800);

        _ = discord.edit_message(
            channel_id.into(),
            placeholder.id.into(),
            &serde_json::json!({
                "content": resp[0]
            }),
        ).await;

        if resp.len() > 1 {
            for resp in resp.iter().skip(1) {
                _ = discord.send_message(
                    channel_id.into(),
                    &serde_json::json!({
                        "content": resp
                    }),
                ).await;
            }
        }
    }
}
```

Step 4: Ask ChatGPT to answer the question based on the context with relevant sections from the book.



SECOND
STATE

Thanks !