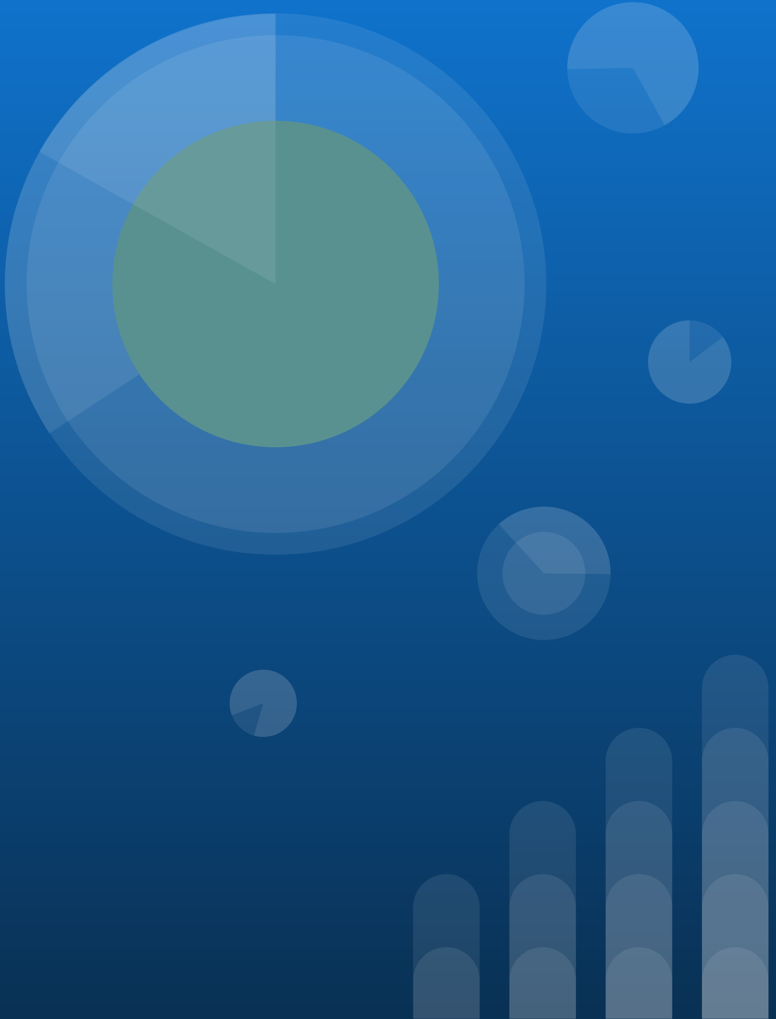
Software Analytics Insights

Final Presentation

Dao McGill, Nicholas Beydler, Mark Burgess, Raven Quiddaoen



Kaiaulu | An R package for mining software repositories

- ➢ An open-source R package for mining software repositories
- ➢ Helps perform data analysis on software development artifacts (gitlog, mailing lists, files, etc.)
- ➢ First created by Carlos Paradis as part of a PhD dissertation in the ICS program

https://github.com/sailuh/kaiaulu

Problem Statement

Understanding Kaiāulu

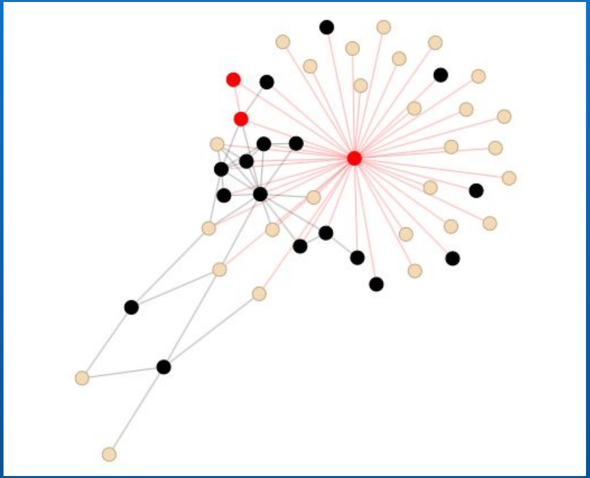
➢ Kaiāulu is an open source R package designed for mining software repositories to analyze their characteristics, by examining collaborative artifacts such as gitlogs, mailing lists, and files.

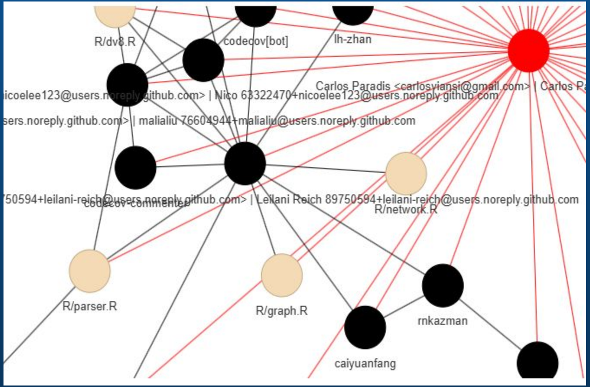
Goal

➢ To refactor and extend Kaiāulu's existing functionality, improving modularity, and introducing more tools for software repository analysis.

Motivation

➢ Software repository analysis tools typically offer features tailored to specific types of analysis with restrictive user interfaces. Kaiāulu aims to provide a flexible user interface composed of generalized functions, with function documentation and notebooks that provide users with abundant examples to customize their analyses.




R/dv8.R
codecov[bot]
lh-zhan
Carlos Paradis <carlosviansi@gmail.com> | Carlos Pa
icoelee123@users.noreply.github.com> | Nico 63322470+nicoelee123@users.noreply.github.com
sers.noreply.github.com> | malialiu 76604944+malialiu@users.noreply.github.com
750594+leilani-reich@users.noreply.github.com> | Leilani Reich 89750594+leilani-reich@users.noreply.github.com
codecov-commenter
R/network.R
R/parser.R
R/graph.R
rnkazman
caiyuanfang

Technical Solution & Implementation

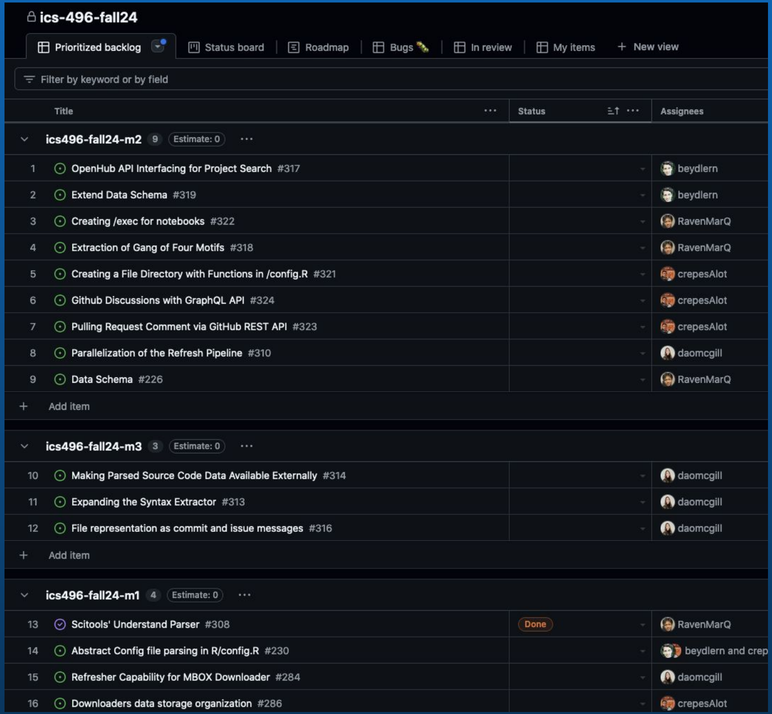
Milestone 1:

➢ Folder organization and config specification:

➢ Getter functions to decouple notebooks

➢ Download and refresh of mailing list archives:
- Developed download and refresh functions for Mod Mbox and Pipermail
- Wrote exec script for download, refresh and parse of mailing list data
- Created notebook for documentation

➢ Extract File & Class Dependencies using third-party tools
- Interact with third-party tool API
- Write function to extract data via API
- Created notebook for documentation



	Title	Status	Assignees
	ics496-fall24-m2 9 Estimate: 0		
1	OpenHub API Interfacing for Project Search #317		beydlern
2	Extend Data Schema #319		beydlern
3	Creating /exec for notebooks #322		RavenMarQ
4	Extraction of Gang of Four Motifs #318		RavenMarQ
5	Creating a File Directory with Functions in /config.R #321		crepesAlot
6	Github Discussions with GraphQL API #324		crepesAlot
7	Pulling Request Comment via GitHub REST API #323		crepesAlot
8	Parallelization of the Refresh Pipeline #310		daomcgill
9	Data Schema #226		RavenMarQ
+	Add item		
	ics496-fall24-m3 3 Estimate: 0		
10	Making Parsed Source Code Data Available Externally #314		daomcgill
11	Expanding the Syntax Extractor #313		daomcgill
12	File representation as commit and issue messages #316		daomcgill
+	Add item		
	ics496-fall24-m1 4 Estimate: 0		
13	Scitools' Understand Parser #308	Done	RavenMarQ
14	Abstract Config file parsing in R/config.R #230		beydlern and crep
15	Refresher Capability for MBOX Downloader #284		daomcgill
16	Downloaders data storage organization #286		crepesAlot

Technical Solution & Implementation

Milestone 2:

➢ Allowing extraction of data using CLI commands
- Read through and understand why, what, and how function(s) output data
- Wrote exec script with relevant data outputs

➢ Analyze projects for usage of Gang of Four Motifs
- Thorough testing of a third-party analysis script with multiple projects and structures to stress test its specificity
- Update existing notebook for improved user clarity

➢ Project selection using OpenHub's database

➢ Extend usage of Github's API endpoints

➢ Create function to create folder directory for configuration files

➢ Parallelize project analysis
- Python parallelization script for adaptation to exec scripts
- Python script for parallelization of mbox parsing
- Jupyter notebook for documentation

Technical Solution & Implementation

Milestone 3:

➢ Expand the syntax extractor
- Functions for file-level docs, class-level docs, classes, variables, functions, imports, packages
- Notebook for syntax extraction
- Exec script to query and parse source code

➢ Use FastText model to generate vector representations of semantic similarity
- Python functions for:
    - Loading model
    - Running queries
    - Tokenization
    - Generating embeddings
    - Calculating cosine similarity and similarity matrix

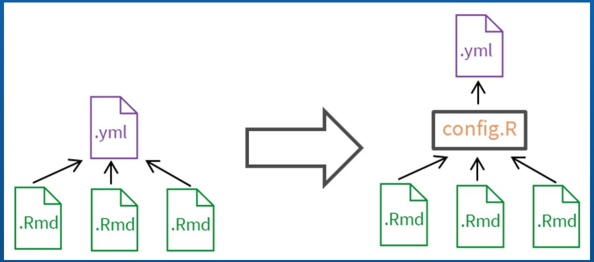
➢ Use UMAP to visualize similarities with dimensionality reduction
- Jupyter notebook for UMAP examples of various representations

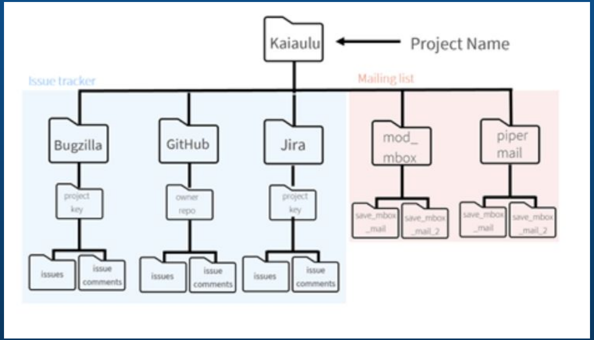
Major Accomplishments (configuration)

➢ Centralized process for gathering details from project configuration (.yml) and **decoupled notebooks (.Rmd)**

➢ Reconfigured **folder organization** for project configuration files at the project level

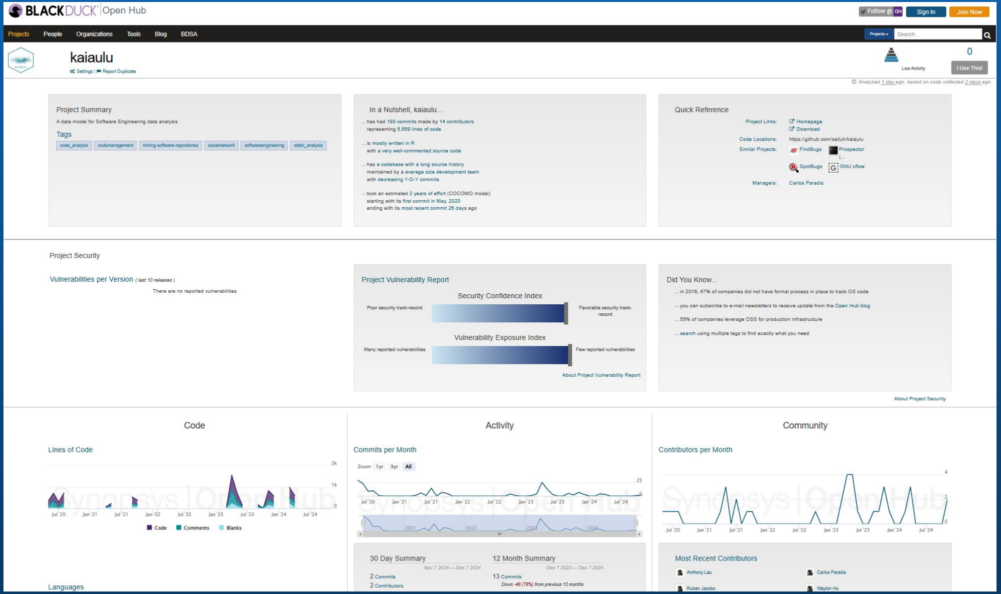
➢ Implemented function to **create folder** organization on user's local device





Major Accomplishments (project selection)

➢ OpenHub API interfacing functions to **select open-source projects** for analysis
➢ Expanded on Github **API endpoints**
➢ Created an interface to **extract dependencies** from source code using Scitools Understand
➢ Extended testing of existing tools to analyze project class files and extract **Gang of Four Motifs**
➢ Implemented download and refresh of two **mailing list data** archives
➢ Python script for the **parallelization** of Kaiāulu functions via exec scripts



kaiaulu
Settings | Report Duplicate

Low Activity
0
I Use This!

Analyzed 1 day ago. based on code collected 2 days ago.

Project Summary
A data model for Software Engineering data analysis
Tags
code_analysis codemanagement mining-software-repositories socialnetwork softwareengineering static_analysis

In a Nutshell, kaiaulu...
... has had 188 commits made by 14 contributors representing 5,659 lines of code
... is mostly written in R with a very well-commented source code
... has a codebase with a long source history maintained by a average size development team with decreasing Y-O-Y commits
... took an estimated 2 years of effort (COCOMO model) starting with its first commit in May, 2020 ending with its most recent commit 26 days ago

Quick Reference
Project Links: Homepage, Download
Code Locations: https://github.com/sailuh/kaiaulu
Similar Projects: FindBugs, Prospector (..., SpotBugs, GNU cflow
Managers: Carlos Paradis

Project Security
Vulnerabilities per Version (last 10 releases)
There are no reported vulnerabilities

Project Vulnerability Report
Security Confidence Index
Poor security track-record — Favorable security track-record
Vulnerability Exposure Index
Many reported vulnerabilities — Few reported vulnerabilities
About Project Vulnerability Report

Did You Know...
... in 2016, 47% of companies did not have formal process in place to track OS code
... you can subscribe to e-mail newsletters to receive update from the Open Hub blog
... 55% of companies leverage OSS for production infrastructure
... search using multiple tags to find exactly what you need

About Project Security

Code
Lines of Code
Code, Comments, Blanks
Languages

Activity
Commits per Month

30 Day Summary
Nov 7 2024 — Dec 7 2024
2 Commits
2 Contributors

12 Month Summary
Dec 7 2023 — Dec 7 2024
13 Commits
Down -48 (78%) from previous 12 months

Community
Contributors per Month

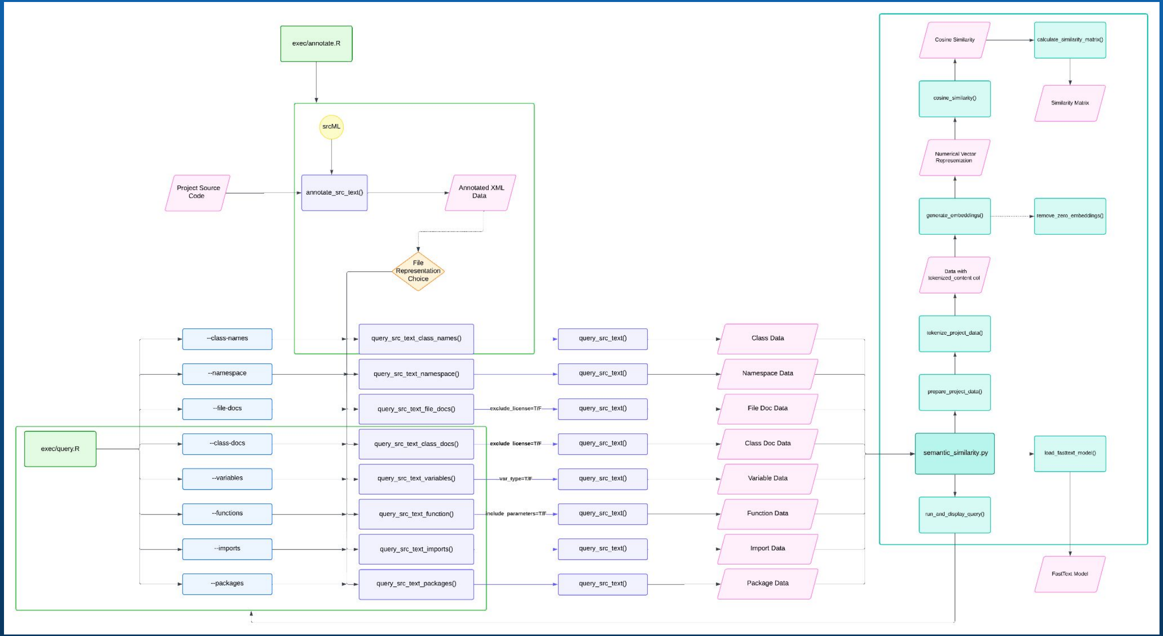
Most Recent Contributors
Anthony Lau
Carlos Paradis
Ruben Jacobo
Waylon Ho

Major Accomplishments (code embedding)

➢ Extended **syntax extraction** capabilities by implementing functions for various elements

➢ Developed **exec scripts** to make parsed source code available to other tools

➢ Used the FastText ML model to generate **code embeddings** for computing **semantic similarities**



Missing/ Partial Implementation

- ➢ Not all functionalities are supported to be used via the CLI
- ➢ Relationships between the outputs of executable scripts have not yet been graphically represented
- ➢ File representations as commit and issue messages have not been implemented

Analyzing Projects : : CHEAT SHEET

Getting Started

Kaiāulu recommends a set of steps to analyze projects. These include the way to organize a project's folders and the way the analysis is configured for reproducibility.

Additional features facilitate project selection by different project demographics, reusing parts of Kaiāulu in other tools or server-side for parallelization.

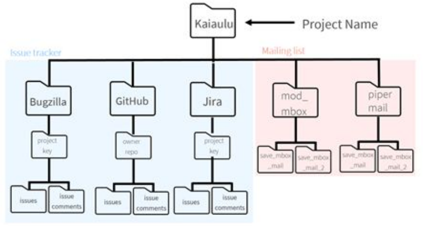
Folder Organization

A project analysis is organized in sub-folders per data source provenance. A project folder can be initialized by the function:

create_file_directory()

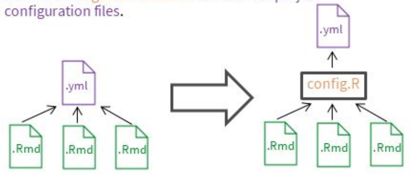
The file organization is specified in the project configuration file. These, in turn, are used throughout the Notebooks via getter functions in Kaiāulu once the project configuration file is specified.

```
project_of_interest                ▪ mailing_list
├── jira                           ▪ issue_tracker
│   └── project_key
│       ├── issue_comments
│       └── issues
├── github
│   └── owner_repo
│       ├── discussion
│       ├── issue
│       ├── pr_comment
│       └── ...
├── bugzilla
│   └── project_key
│       ├── issue_comments
│       └── issues
├── mod_mbox
│   ├── save_mbox_mail
│   └── save_mbox_mail_2
└── pipermail
    ├── save_mbox_mail
    └── save_mbox_mail_2
```



Accessing Configs from Notebooks

The getter functions in config.R centralize the process of gathering information from project configuration files (.yml) in Notebooks (.Rmd). This allows for different Notebooks to use the same getter functions for different project configuration files.



Without the getter functions from config.R, the Notebooks would require direct variable assignments to the project config information.

With the getter functions from config.R, the Notebooks can use the getter functions to acquire the project config information.

If the project config specification evolve in the future, only the corresponding getter function implementation needs to be updated, instead of all the dependent Notebooks and exec scripts.

Selecting Projects

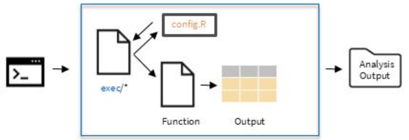
Project selection often involves tedious work in browsing different websites and repositories to identify project demographics, language, issue traceability, or bug labeling.

openhub_project_search.Rmd is a Notebook that showcases how Kaiāulu interfaces with OpenHub API to facilitate selecting open-source projects for studies. This Notebook demonstrates how to use the Kaiāulu OpenHub API interface to search for projects that meet specific criteria, streamlining the process of discovering open-source projects through OpenHub's database.

Once projects for analysis are decided upon, they can be documented as project configuration files, in turn accessed in Notebooks by getter functions.

Command Line Interface

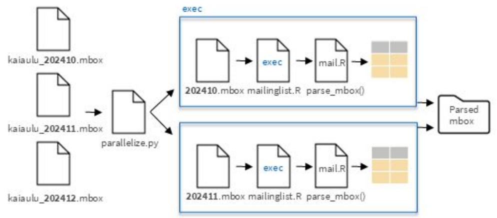
Exec scripts serve as interface between Kaiāulu functions and external tools, allowing a subset of capabilities to be accessed via the command line in other languages.



Exec scripts also utilize getter functions to access project configuration files, and Kaiāulu's API to externalize functionality. Exec scripts expect a combination of both project configuration files as input and optional arguments.

Parallelization

Through exec scripts, Kaiāulu functions can be called in parallel, enabling concurrent analysis of large projects.



The parallelize.py script can be used with other Kaiāulu exec scripts depending on the analysis needs.

Kaiāulu

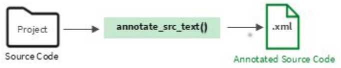
File Similarity by Code Representation : : **CHEAT SHEET**

About

Files can be represented differently depending on how we subset their content, such as functions, class names, variables, or documentation. These representations shape the way files are similar to one another, as selected code snippets emphasize different semantic relationships between files. This cheat sheet describes how to use **Kaiāulu** to represent source code differently for semantic similarity.

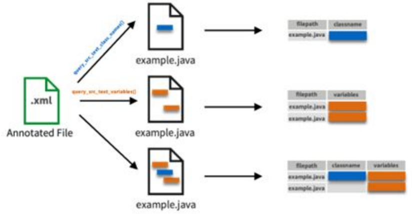
Source Code Representation

The Syntax Extractor in Kaiāulu is used to extract selected content from source code using srcML. The srcML tool is used to annotate source code with descriptive tags that help in identifying code elements.



annotate_src_text(): generates an annotated XML file.

XPath queries are used to query the annotated file, and output a table of filenames and the source code element being queried (variable names, function names, file class documentation, etc.)



filepath	classname
example.java	🟦

filepath	variables
example.java	🟧
example.java	🟧

filepath	classname	variables
example.java	🟦	🟧
example.java		🟧

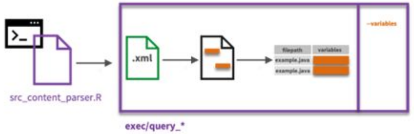
Query functions: Kaiāulu provides an API for extracting different portions of source code files as structured tables. These representations include variable names, function names, documentation, and more, each capturing specific file elements. Using these tables, we can analyze and transform code into selected representations for further processing.

query_src_text_class_names(): outputs a table of with a class names column, and a file paths column.
query_src_text_namespace(): outputs a table with a namespaces, and a file paths column.

Relevant Notebook: syntax_extractor.Rmd

Exec Scripts

Exec scripts act as interfaces between Kaiāulu functions and external tools, exposing Kaiāulu's capabilities to external sources through the command line. Users can execute scripts for specific queries (e.g., extracting function names or documentation) from the terminal.



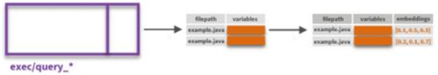
filepath	variables
example.java	🟧
example.java	🟧

src_content_parser.R: serves as the main entry point for querying. It takes a flag specifying the query type (e.g., --class-names, --variables) and calls the associated query function. The annotated file is passed as a parameter, determining the contents to be queried.

query_* scripts: implement the logic for specific types of queries, such as extracting class names or variables. Each query focuses on a particular aspect of the code representation.

Embeddings

Word embeddings represent text semantics by mapping words or sentences to a numerical representation. The numerical vector representation can be used for semantic similarity comparisons which would be lost if direct word comparison was utilized.



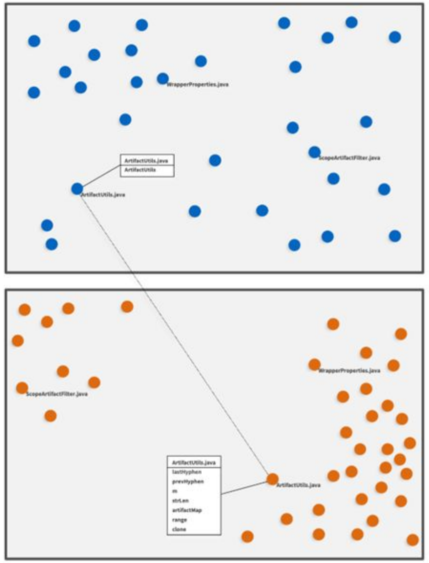
filepath	variables
example.java	🟧
example.java	🟧

filepath	variables	embeddings
example.java	🟧	[0.1, 0.3, 0.3]
example.java	🟧	[0.2, 0.1, 0.7]

Beyond the selection of a file representation, different models to convert content to embeddings can also be selected. For example, Java-specific FastText embeddings may be better for source code representations, while a file representation containing only code documentation may benefit from natural language embeddings instead of source code. Other language models may also be more appropriate if the source code is not Java. Kaiāulu parameterizes the pipeline.

UMAP

UMAP transforms high-dimensional embeddings by reducing dimensionality and fitting on a plane, which allows for visualization of the relationships between files.



The diagrams above show how the representation of code elements, such as class name and variables, impacts the relative positions of files in a UMAP projection. Each representation emphasizes different aspects of the code, resulting in varied groupings and relationships.
This variability demonstrates the importance of selecting the appropriate representation for the specific insights required. By choosing different code elements, similarity analysis can be tailored to answer diverse research questions.

Kaiāulu

Sponsor Feedback

- “The team performed an immense amount of high quality work throughout Kaiaulu codebase, not only in adding new features, but also refactoring major parts of Kaiaulu to more easily accommodate new changes. I appreciate the engagement, enthusiasm, and availability of the group in seeing the work through. – Carlos”

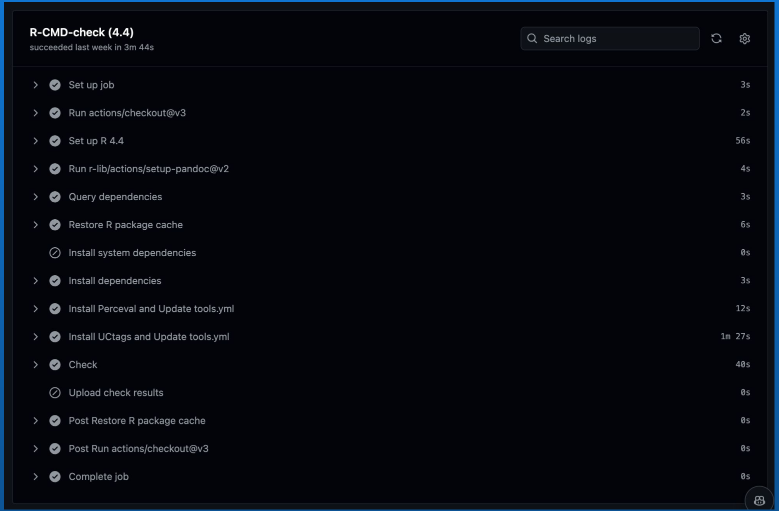
Quality Assurance

➢ Our Quality Assurance Plan aids us in verifying our code for defects.

➢ Code undergoes local checks and units tests, and commits are checked by GitHub Actions.

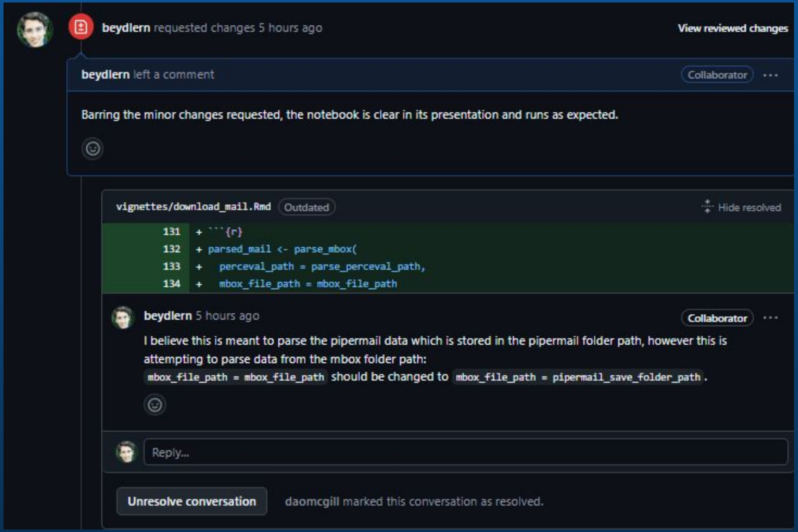
➢ Our code is reviewed by peers through code review requests, and by our sponsor through pull request reviews.

➢ This review process ensures that reported defects are fixed once spotted and the review comment pointing the defect out is resolved to keep track of closing defect reports.



```
R-CMD-check (4.4)
succeeded last week in 3m 44s

Set up job                                   3s
Run actions/checkout@v3                      2s
Set up R 4.4                                56s
Run r-lib/actions/setup-pandoc@v2            4s
Query dependencies                           3s
Restore R package cache                      6s
Install system dependencies                  0s
Install dependencies                         3s
Install Perceval and Update tools.yml       12s
Install UCtags and Update tools.yml     1m 27s
Check                                       40s
Upload check results                         0s
Post Restore R package cache                 0s
Post Run actions/checkout@v3                 0s
Complete job                                 0s
```



beydlern requested changes 5 hours ago

beydlern left a comment

Barring the minor changes requested, the notebook is clear in its presentation and runs as expected.

vignettes/download_mail.Rmd Outdated

```
+ ```{r}
+ parsed_mail <- parse_mbox(
+   perceval_path = parse_perceval_path,
+   mbox_file_path = mbox_file_path
```

beydlern 5 hours ago

I believe this is meant to parse the pipermail data which is stored in the pipermail folder path, however this is attempting to parse data from the mbox folder path:
`mbox_file_path = mbox_file_path` should be changed to `mbox_file_path = pipermail_save_folder_path`.

daomcgill marked this conversation as resolved.

Project Management and Delivery

➢ Our project management approach is well-established and has remained consistent from the start.

Task Management Tracking:

➢ Github is our task management tracker (assigned issues).

➢ Issue-based PRs to keep track of progress, and allow for easy code review.

Project Artifacts:

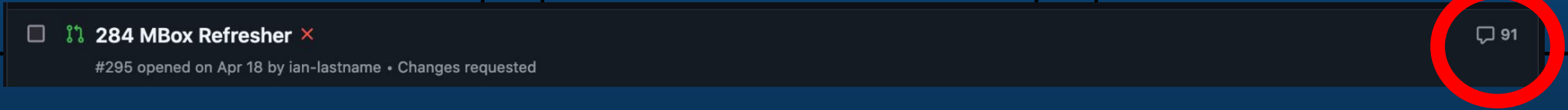
➢ The creation of notebooks, the functions we implement, and the executable scripts we develop are the primary coded project artifacts that aim to display the new functionalities introduced to Kaiaulu.

Quality Assurance Plan:

➢ Our code is peer reviewed and reviewed by our project sponsor.

➢ Github Actions and test cases for automated code testing and building.

➢ Consistent communication and iterative feedback within issues and PRs.



284 MBox Refresher ✕

#295 opened on Apr 18 by ian-lastname • Changes requested

91

Reflections

Success Highlights

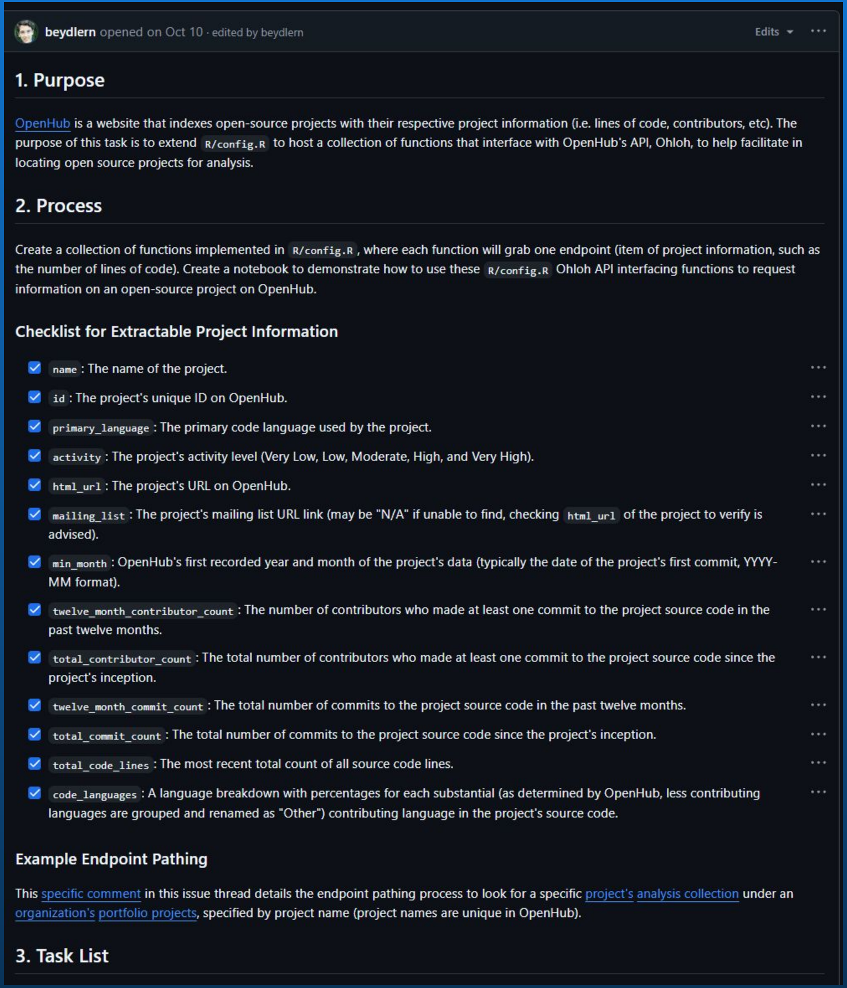
➢ Collaborating with our sponsor to define requirements through issue specifications helped to refine the scope of our tasks to mitigate scope creep.

➢

Areas for Improvement

➢ Documenting messages and conversations under relevant issues

➢ More frequent peer code reviews



beydlern opened on Oct 10 · edited by beydlern

Edits ▾ ···

1. Purpose

OpenHub is a website that indexes open-source projects with their respective project information (i.e. lines of code, contributors, etc). The purpose of this task is to extend `R/config.R` to host a collection of functions that interface with OpenHub's API, Ohloh, to help facilitate in locating open source projects for analysis.

2. Process

Create a collection of functions implemented in `R/config.R`, where each function will grab one endpoint (item of project information, such as the number of lines of code). Create a notebook to demonstrate how to use these `R/config.R` Ohloh API interfacing functions to request information on an open-source project on OpenHub.

Checklist for Extractable Project Information

- [x] `name` : The name of the project.
- [x] `id` : The project's unique ID on OpenHub.
- [x] `primary_language` : The primary code language used by the project.
- [x] `activity` : The project's activity level (Very Low, Low, Moderate, High, and Very High).
- [x] `html_url` : The project's URL on OpenHub.
- [x] `mailing_list` : The project's mailing list URL link (may be "N/A" if unable to find, checking `html_url` of the project to verify is advised).
- [x] `min_month` : OpenHub's first recorded year and month of the project's data (typically the date of the project's first commit, YYYY-MM format).
- [x] `twelve_month_contributor_count` : The number of contributors who made at least one commit to the project source code in the past twelve months.
- [x] `total_contributor_count` : The total number of contributors who made at least one commit to the project source code since the project's inception.
- [x] `twelve_month_commit_count` : The total number of commits to the project source code in the past twelve months.
- [x] `total_commit_count` : The total number of commits to the project source code since the project's inception.
- [x] `total_code_lines` : The most recent total count of all source code lines.
- [x] `code_languages` : A language breakdown with percentages for each substantial (as determined by OpenHub, less contributing languages are grouped and renamed as "Other") contributing language in the project's source code.

Example Endpoint Pathing

This specific comment in this issue thread details the endpoint pathing process to look for a specific project's analysis collection under an organization's portfolio projects, specified by project name (project names are unique in OpenHub).

3. Task List

Reflections

Possible Adjustments

➢ To establish clearer task distribution earlier on

➢ Updating code to consolidate changes from individual tasks

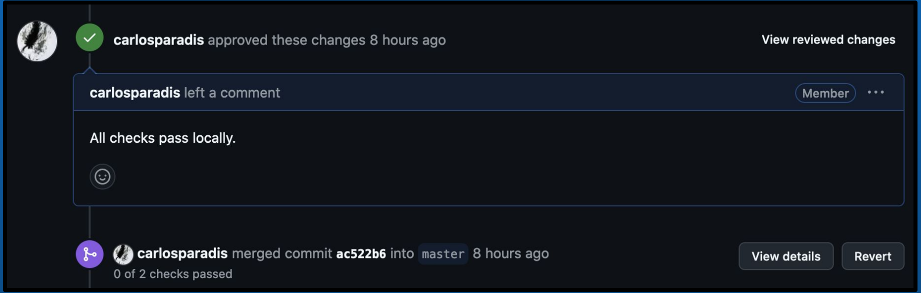
Learned and Strengthened Skills

➢ Agile project development

➢ R programming language and RStudio (IDE)

➢ Thorough documentation process of our implementations

➢ The review process using GitHub



carlosparadis approved these changes 8 hours ago

carlosparadis left a comment

All checks pass locally.

carlosparadis merged commit ac522b6 into `master` 8 hours ago

0 of 2 checks passed

Next Steps

Group

➢ Ensure code standard and project specifications are met through pull request code reviews.

Sponsor

➢ Perform a review of our Pull Requests and merge our code to the master branch.

Acknowledgements

Carlos Paradis

➢ Stakeholder/Sponsor/Mentor, Ph.D. in Computer Science from UH Manoa

Rick Kazman

➢ Sponsor, Professor of Information Technology Management at UH Manoa