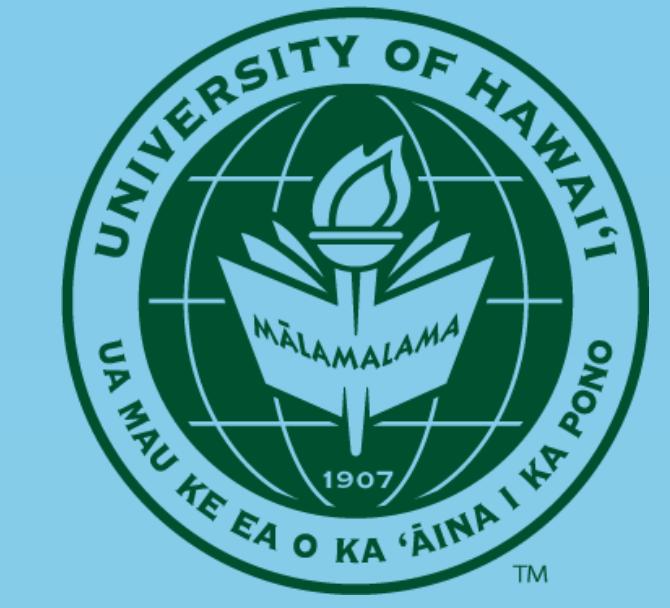


Sentiment Analysis and Kaiāulu React

Austin Dang, Gerald Matthew Huff
 Sponsor: Carlos Paradis, Software Analytics Insight Lab
 ICS 496 Capstone Project – Spring 2025



github.com/sailuh/kaiaulu



What is Kaiāulu?

Kaiāulu is an R package and common interface that helps with understanding evolving software development communities, and the artifacts (gitlog, mailing list, files, etc.) which developers collaborate and communicate about which allows developers to get a better understanding of the software.

Objectives

- Sentiment tools:** Introduce sentiment-analysis tools for project communication data using machine-learning models to help researchers and developers understand team emotional tone and communication trends.
- Preprocessing improvements:** Improve data preprocessing methods to ensure necessary communication data is consistent and ready for accurate model training and sentiment prediction.
- Web application for software analytics:** Create a web application for software analysis by providing an interactive interface for visualizing data extracted by Kaiāulu.
- Scalable web architecture:** Establish a scalable architecture that supports the efforts of future developers in building upon the Kaiāulu web application.

Challenges

- Understanding diverse communication data was challenging; emails, comments, and issue trackers store information differently, requiring tailored preprocessing to filter out irrelevant text so the ML model can focus on meaningful content.
- Implementing a consistent architecture when adding new features to the existing Kaiāulu repository, requiring careful adherence to the established workflows and design patterns.
- Designing a robust architecture for building a web application that will be understood and expanded upon by future contributors.
- Utilizing a JavaScript data visualization library to create a compelling interface for software analysis that supports the capabilities of the Kaiāulu R package.

Tasks accomplished

- Built a preprocessing pipeline for cleaning and standardizing communication data (emails, Github comments, Jira issues)
- Configured project workflows using standardized YAML files for reproducibility.
- Integrated Python-based ML scripts into R notebooks to perform sentiment analysis, creating a reusable method that enables future users to assess overall sentiment in new datasets and better understand developer communication trends.
- Created a web application that utilizes data extracted by the Kaiāulu R package and provides several features to assist developers in analyzing software.
- Created documentation that will inform new contributors of the logic behind existing features and how to expand upon the application while maintaining proper organization.

Kaiāulu React: : CHEAT SHEET

About

This cheat sheet showcases Kaiāulu React's capabilities in displaying project painpoints with communication, collaboration, and code design.

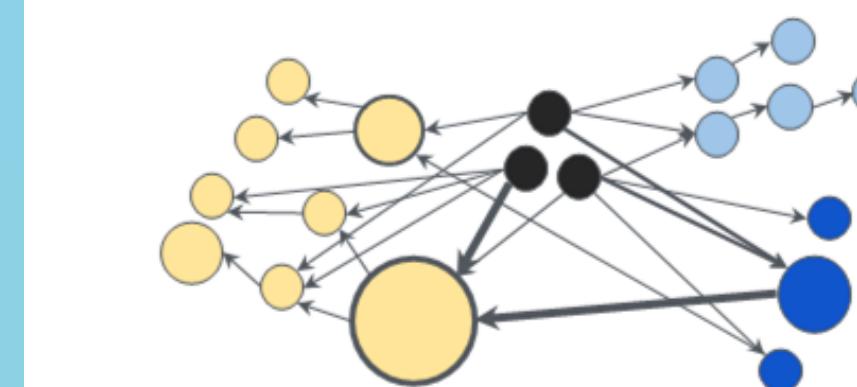
Configuration File

Kaiāulu React requires one or more networks of code artifacts (files), communication (mailing list), issue tracker (e.g. JIRA, GitHub), and developers (e.g. authors, committers).

The data can be mined using sailuh/kaiaulu, and its paths are specified on sailuh/kaiaulu/react/kaiauluyml configuration file. Users can version different project configurations for easy swaps.

Landing Page

The landing page features an interactive Network Graph which displays collaboration, communication, and code dependency relationships.



The project of interest is displayed as nodes and edges. Users can choose any node via Kaiāulu to represent the weight and label of both nodes and edges, reflecting their size.

By illustrating weight visually, developers can immediately identify significant or potentially problematic areas of their project.

Kaiāulu

Data Types

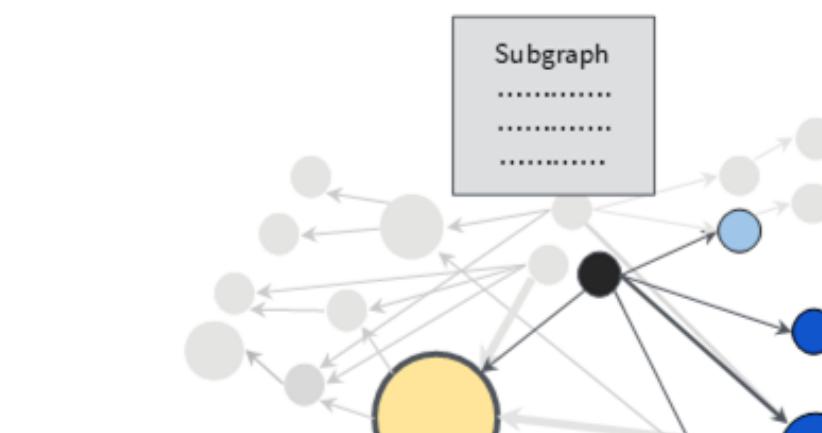
Data may contain one or more of:

- People – the developers.
- Files – the code artifacts.
- Mail – email replies.
- Issues – issue comments.

Users may also select other equivalent data types (e.g. Discord channel) if they use the same data format, as the metrics are defined over the data.

Subgraph Highlight

Developers may select nodes of interest and create a highlighted subgraph that is accompanied by a panel containing further information about the nodes and relationships present in the subgraph.



Developers may also search for specific nodes and set filters to refine the data shown on the Network Graph.

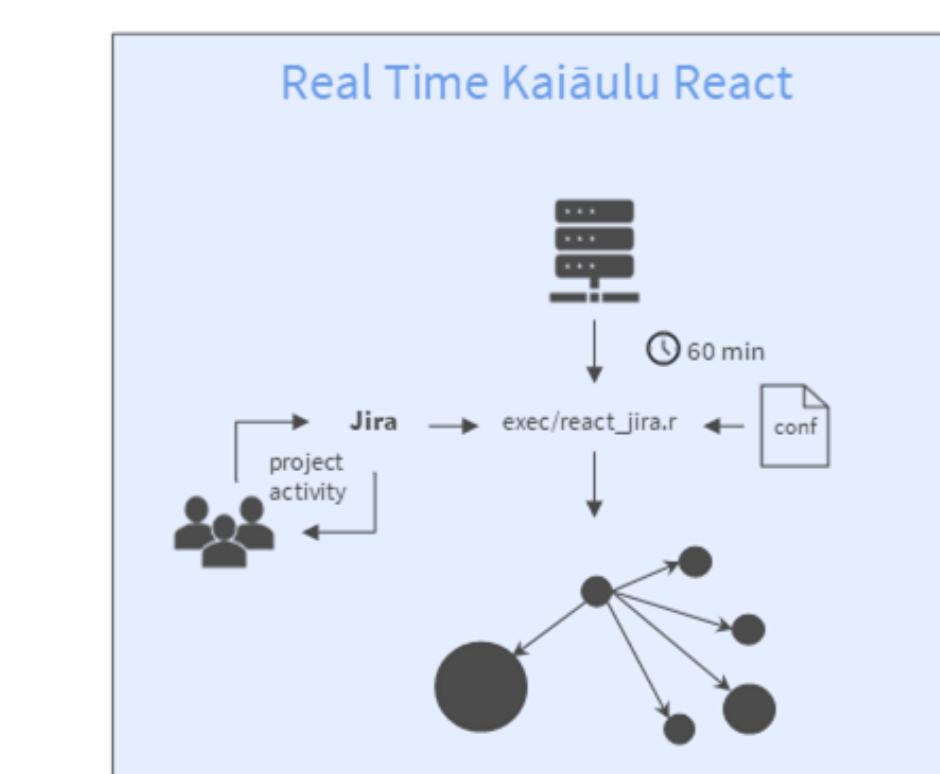
Enter node label Search Filters v

Example

Consider the following network graph...

 A file node sized by its lines of code... Person B
 Person A
 Person C
 ... implementation of the file.
 Developers can immediately identify parts of the project that warrant additional attention by visual weight, which can be set to scale nodes and relationships by metrics of interest.

e.g. "this file has a lot of code and... it appears that Person A is a major contributor, but they are not a part of the discussion on the issue."
 or... "this file has a lot of dependencies. Could we perhaps split up it up and separate its concerns?"



Sentiment Analysis: : CHEAT SHEET

About

Kaiāulu supports the download of comments from JIRA, GitHub, Mailing Lists and more. This cheat sheet showcases Sentiment analysis which classifies a author's comment as negative, neutral or positive. Sentiment analysis can be used to assess unhealthy communication patterns in a project, potentially affecting its code quality.

Project Config Setup

The first part of running any vignette is setting up your project configuration file (examples in config/kaiaulu).

Required Fields	
filter:	replies: msg_by_reply_author_substring: filter_by_reply_subject_substring: filter_by_reply_body_substring: regex_to_replace_key_with:
mailing_list:	mod mbox: project_key_1: project_key_2: project_key_3:
issue_tracker:	jira: project_key_1: issue_comments: github: project_key_1: issue_or_pr_comment:
tool:	sentiment: model: prediction:

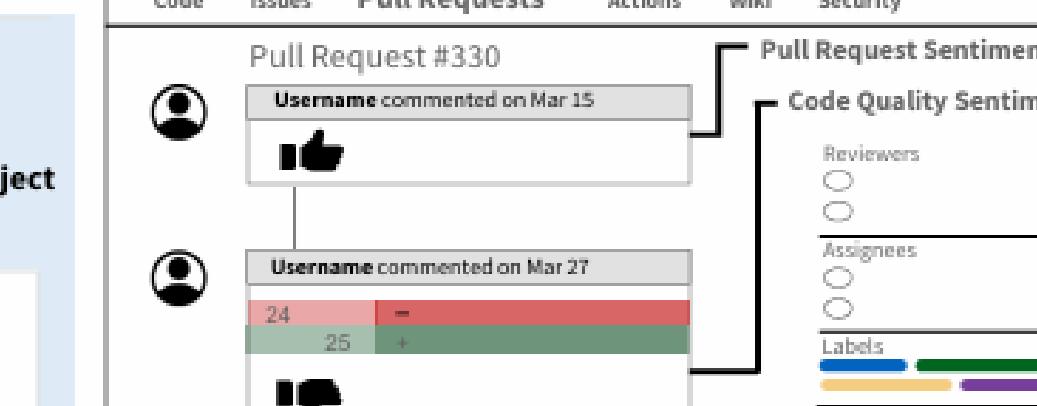
You should also specify the required external tools in tools.yml:

Tools Config Setup	
Required Fields	
perceval:	pysenti:

Kaiāulu

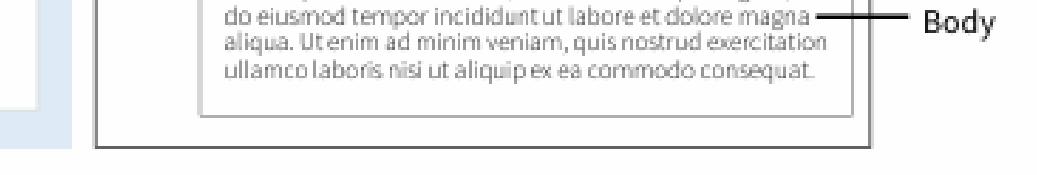
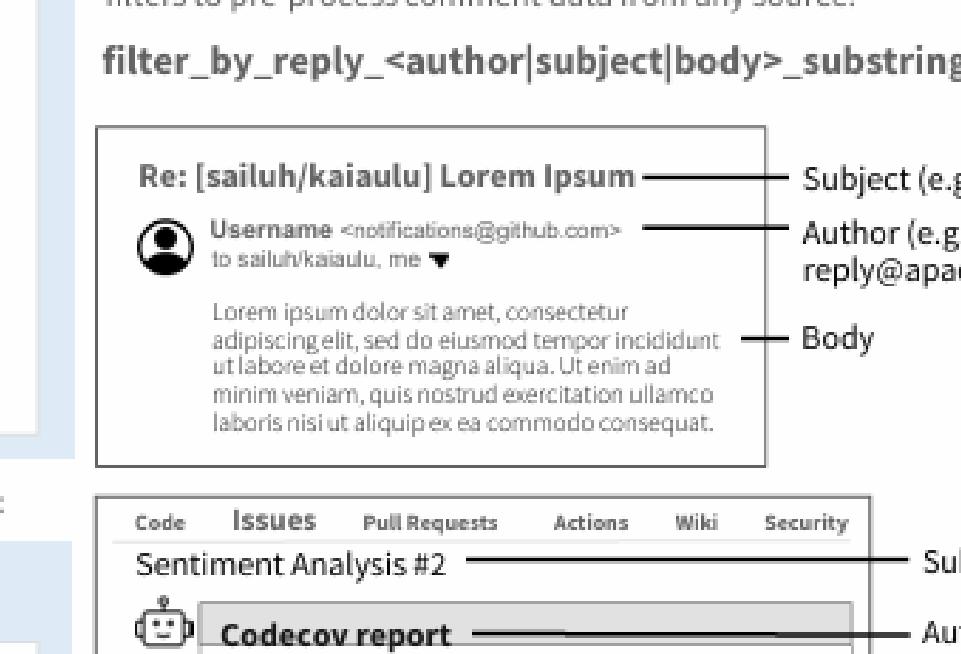
Comment's Sentiment

Comments can occur in different data sources (GitHub, Jira, Mailing list, and also occur under different views within the same data source). Sentiment labels carry different meaning depending on the view (e.g. feature communication, specific code line review). Sentiment can also be extracted either from text or emoji.



Reply Filters

Developer comments generally include code blocks, automated messages, bot messages, markdown formatting, etc. To ensure that the model focuses only on human-generated content, Kaiāulu offers filters to pre-process comment data from any source.



replace_token_regex_with()

Replaces text matching given regex patterns with standardized token placeholders (e.g. MURL replaces <https://example.com>, MEMAIL replaces user@example.com, etc.)

filter_text_punctuation_markdown_newlines_whitespace()

Text-cleaning functions that remove text content and convert markdown to plain text.

pySENTI_train_model(pysenti_path,...)

Trains a specified model using (manually) labeled sentiment data, and saves the weights to the specified project configuration model path.

f [W1, W2, ..., Wr]

pysenti_predict(pysenti_path,...)

Lays a saved model weights to classify the sentiment of a data table's unlabeled comments and returns a data table with comment's classified sentiments.

Technologies

- Git & GitHub
- Languages: TypeScript + CSS, Python, R
- JavaScript Libraries: React, [D3.js](#), MUI
- Third Party Tools: Perceval, Pysenti

Learnings & Conclusions

The project reinforced the importance of adhering to Kaiāulu's architecture. By implementing reproducibility methods within preprocessing steps, it can be adapted to different communication sources which supports flexible sentiment-analysis workflows that can be reliably applied across projects and makes the overall process easier to manage.

Creation of the Kaiāulu web application from the ground up allowed for a strong reinforcement of development skills in addition to a newfound understanding of architectural design. The rippling effects of design choices made at the beginning of this project was an invaluable learning experience. Furthermore, deep insight was gained when designing an application with developers in mind and even more regarding data visualization.