Collaborations Workshop 2019 (CW19) #CollabW19

2019-04-01 to 2019-04-03

Team 7 - CI5-CW2CC

Reporter

Sam Mangham - mangham@gmail.com

Participants

Alexandra Simperler, alex@simperler-consulting.com,
Mario Antonioletti - mario@epcc.ed.ac.uk,
Sam Mangham - s.w.mangham@soton.ac.uk,
Ben Krikler - b.krikler@cern.ch,
Alejandra Gonzalez-Beltran - alejandra.gonzalez.beltran@gmail.com,
Tyler Whitehouse

This document should be used to capture the information for a Collaborative Session / Hack Day Idea. (The total amount of text should ideally be between 100-300 words and you can include a diagram or two). The document should be no larger than two pages of A4. Don't delete the details at the top of the document but you should delete all the hint text once you no longer need it.

Context / Research Domain

Developers in any research domain that want to find appropriate libraries (this could be extended to other resources).

Problem

Developers can find it hard to identify suitable software libraries for their work. Given a problem, deciding on which libraries to use is difficult; even if you know one, it's hard to find other libraries that do the same thing, or equivalent libraries in other languages. What is the approach with the most traction, particularly within a given community? Which libraries are popular, but low-quality? Serendipitous discovery of libraries is also particularly hard.

Solution

Building a recommendation system that incorporates feedback, using information from similar individuals, and what packages are present (what signal) in your GitHub repo. Based on these, a NetFlix-style recommendation system could suggest possible libraries, and you could give feedback on how suitable a suggested library is for your application. With sufficient iteration, it would be possible to build up a profile of your tastes or needs and how they relate to other users. This would also make it possible to aid discovery by recommending libraries that other

similar developers use.

A decentralised, user-maintained business model might be easier to sustain. As a result, an initial implementation could work by providing a script or tool that queries existing package and library aggregators and rankers. A user would configure this tool by cloning a template repository from the official github, and setting it to run regularly within a continuous integration pipeline. Specific customisations like types of packages, licensing, etc, is left in the configuration of the user's cloned repository. Additionally, automated summaries of the user's code (styles, typical package uses, languages, etc) can be produced using the developed tools and stored in the user's repository. Recommender functionality that looks at collaborator's styles and dependencies can work by querying colleague's public clone of the same repository.

Diagrams / Illustrations

Preliminary Notes

- What do people want from an open simulation platform interface? Interoperability requirements
    - Identifying user needs
- Software scouting- increasing accessibility/discoverability for tools.
    - Registries? They tend to go out of date.
    - Ways to recommend solutions
    - Can you de-centralise a registry?
        - Peer-to-peer registries?
        - Deciding on a vocabulary, tagging pages a la Schema.org
        - Not a lot of scientific content on it
        - Encourage users to adopt this existing standard to aid discoverability via Google/existing tools
        - Schema.org
        - One person lingua franca is another person's dying language
    - Determining what tools/techniques/sites have the most traction, and focus on them. Positive feedback loop.

        - Use big names/brand recognition to break into new fields, rather than relying on ‘if you build it they will come’.
        - ‘Metrics for movements’ is the hard part. Current use vs new adoption. Difficulty of selecting easy-to-calculate/search on metrics
        - GitHub starring/cloning etc. metrics exist, but may not be advertised.
        - Libraries.io does this a bit
        - Proper, visible metrics BUT avoiding information overflow.
        - Concise, interactive, easy-access information
        - Problem: Popularity != quality. Differentiate between popular but low-quality content and popular good content. How?
            - Automated (e.g. code quality/unit testing)
            - Fair Sharing standards for events etc.
            - Matching tools & metrics e.g. like booking.com/dating sites!
            - (similar: Matching RSEs to work)
- Service on top of the github api that provides different metrics or indicators
    - You get to choose the metrics so that you can make sure results are per an “educated choice”
- Tag libraries & resources to make it easy to find similar ones and compare the quality/popularity
- Preferential attachment - high degree nodes attract more connections; it can work poorly
- Educated choice - a service that would give you all the metrics that allow you to make a choice
- Daily newsletter (i.e. libraries.io generated) that gives you what you know but also let’s novel things bubble up into your feed (satisfies need and bias but presents things that are “orthogonal” to your perspective)
    - Promotion of new content that is seeing a substantial increase in popularity. Threshold for speed of adoption -> promotion.
- Recommendation service for libraries.
    - Follow people on github to get repository starring information.
    - Curators? Serendipitous discovery by following people and discovering the stuff they use.
    - Dating-service-like feedback on quality of recommendations.
- “Tell me your code and I’ll tell you who you are”
- Modify recommendations with jitter/noise to escape echo-chamber