Mining Software Repositories

Report

Presented By Team: Oscar

Introduction

- Idea of reusability is intensely utilized in the present programming advancement.
- Less time and exertion is expected.
- GitHub repositories.

Total Java Repositories on GitHub	10 Million
Repositories with at least 100 stars	19000
Repositories with 100 stars, two contributors, 100 commits, and one POM file.	4018
Parsable repositories	3778

Methodology of the Thesis

- In the first step, GitHub repositories are gathered, selected, and filtered according to their disclosed dependencies.
- Following that, the chosen repositories are processed and searched for API usages.
- After that, the parsed repositories and API usages are examined using a categorization and sample approach.
- Finally, treemaps are used to visualize the repositories and their API usage.

Research Question

- ***“Can we continuously sample projects and aggregate knowledge about combined API usage?“***

- As we know that Parsing huge projects is very costly and we know that if we can somehow select projects with high usage to cover cleverly, then this may be a big performance win.

- Once we have seen a few projects (perhaps just 1) with actual usage of API A and B, but without any significant combined usage (in methods), we may start focusing on projects that exercise other APIs. Once we have seen “all” APIs and pairs, we may start more iterations over the APIs.

Potential Solution

- **Our Idea:**

➔ In the thesis methodology, First the dependency pairs are being created and then only those repositories are parsed which mention the dependency pair in their POM file.

➔ Secondly, if we skip a dependency pair just after parsing initial few repositories and then resume the iterations after all other dependency pairs have been analyzed. Then ultimately, we end up doing more processing than the current solution.

➔ If we have to resume the parsing in the end then this might not be helpful in saving processing costs.

Potential Solution

- **Additional opinions:**

  Additionally, before starting the analysis of the repositories, we may first perform a validation check that if there are any dependencies mentioned in the POM file but they are not used in the actual code then skip those projects, then it might save some processing costs.

- **Final words:**

  As far as the research question is concerned, despite changing various attributes (including number of repository stars and number of contributors) in the repository selection step,
  we were not able to find any optimizations which would reduce the processing costs without giving up important information.

Thank You!