DISCOVERING DIGITAL SIBLINGS

QUANTIFYING INTER-REPOSITORY SIMILARITY THROUGH GITHUB DEPENDENCY STRUCTURES

Author
Mateusz Rębacz
m.rebacz@student.tudelft.nl
Responsible professor
Sebastian Proksch
Supervisor
Shujun Huang

MOTIVATION

Digital siblings - *Repositories with similar goals or problem domains.*

- The Open Source software ecosystem is extensive, allowing developers to collaborate, share and learn from the code of others.
- Finding repositories similar to one's own allows for better collaboration, knowledge transfer, and code reuse.
- With more than 100 million repositories on GitHub, finding similar projects manually is a very difficult task.
- Dependencies, such as libraries and frameworks used by software projects, provide insight into the project's topic/problem domain, making an automated approach for finding digital siblings possible.

RESEARCH QUESTIONS

How can the dependency structures of GitHub repositories be leveraged to find their digital siblings?

- **RQ2:** What **metrics**, derived from analyzing dependency structures, most accurately quantify the similarity between GitHub repositories?
- **RQ3:** Which **clustering methods** are most effective in grouping GitHub repositories into clusters mirroring similar problem domains?
- **RQ4:** How can dependency structures as a similarity metric be **composed** with the similarity metrics investigated by the other RP group members?

METHODOLOGY

Data Collection - We collect a list of repositories to analyse.
Dependency Extraction - Direct and transitive dependencies are extracted automatically from each repository in our dataset.
Repository Vectorization - Each repository is transformed into a binary vector representing the dependencies used by the project.
Performance Evaluation

- **Similarity Metrics** - Euclidean Distance, XOR Sim., and AND Sim.
- **Clustering Techniques** - Agglomerative, K-Means, and DBSCAN.
- **Composable Approach** - Most effective similarity metric and clustering technique combined for a hybrid, composable metric.

DATA COLLECTION

```
# minecraft plugins
https://github.com/filoghost/HolographicDisp
https://github.com/NoCheatPlus/NoCheatPlus
https://github.com/ViaVersion/ViaBackwards
https://github.com/games647/FastLogin
https://github.com/TownyAdvanced/Towny
https://github.com/SkinsRestorer/SkinsRestor
```

Dataset: Selecting a list of GitHub repositories to analyse.
Currently: Minecraft modifications vs. server plugins.

DEPENDENCY EXTRACTION

```
  }
 ]
},
{
 "provider": "github.com",
 "repo": "AuthMeReloaded",
 "user": "AuthMe",
 "dependencies": [
   {
    "name": "javax.xml.bind:jaxb-api",
    "version": "2.3.1"
   }
 ]
},
{
 "provider": "github.com",
 "repo": "Slimefun4",
 "user": "Slimefun",
```

Extraction: Extracting dependency tree information for each repository in the data set.

REPOSITORY VECTORIZATION

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ \dots \end{bmatrix}$$

org.spigotmc:spigot-api
com.github.seeseemel....
junit:junit
github.scarsz:configur...
com.vdurmont:emoji-...

Algorithm Design & Implementation: Creating a similarity metric based on shared dep. structures.

1. SIMILARITY METRICS EVALUATION

EUCLIDEAN DISTANCE

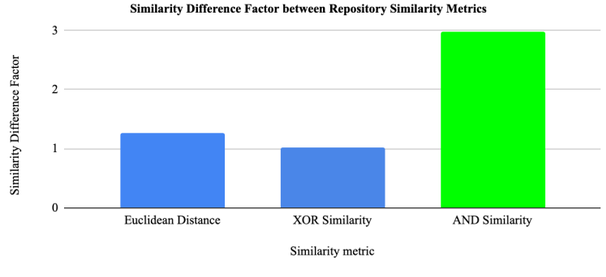
Calculates the Euclidean distance between repository dependency vectors.

XOR SIMILARITY

Performs XOR between repository dependency vectors and uses the proportion of matching dependencies as a similarity metric.

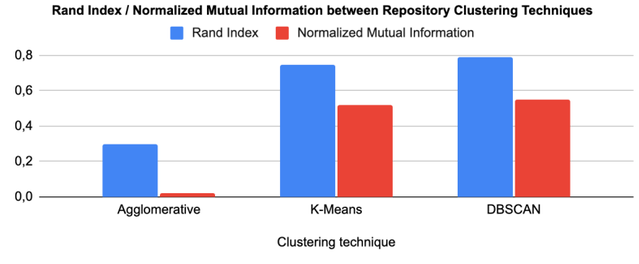
AND SIMILARITY

Performs AND between repository vectors and uses the proportion of shared dependencies as a similarity metric.



2. CLUSTERING TECHNIQUES EVALUATION

AGGLOMERATIVE K-MEANS **DBSCAN**



3. COMPOSABLE SIMILARITY METRIC

Combines *Similarity Metrics* with *Clustering Techniques*.
A model M is trained by computing k clustering using the chosen training method. In inference, a characteristic vector C is computed for each unseen vector by calculating the distance of the vector to each cluster in pre-trained M. Similarity is then calculated using Euclidean distance on characteristic vectors.

CONCLUSIONS

RQ2: Similarity Metrics
The *AND Similarity* metric proved the most efficient. On average, it assigns ~3x higher similarity values when repositories are *digital siblings*, as opposed to dissimilar repositories.
RQ3: Clustering Techniques
The *DBSCAN* clustering proved the most efficient. ~79% of clusters produced by DBSCAN aligned with the reference clusters.
RQ4: Composable Approach
Training the model using K-Means (k = 5) clustering proved the most efficient. The metric assigns ~2x higher similarity values to digital siblings.
Main question:
There are effective similarity metrics and clustering techniques which can identify groups of digital siblings in a set of GitHub repositories.

4. COMPOSABLE SIMILARITY METRIC EVALUATION