Präferenzen einer Organisation

GitHub[1] ist nicht nur ein weiterer Onlinedienst für die gleichnamige Versionsverwaltungssoftware Git[2], mit dem Entwickler ihre Projekte auf zentralen Repositorien verwalten können, sondern präsentiert sich auch als ein soziales Netzwerk für einzelne Programmierer und ganze Organisationen. GitHub bietet sämtliche Funktionalitäten, die für ein gemeinsames Arbeiten an einem Softwareprojekt benötigt werden. Aus diesem Grund und der Möglichkeit, kostenlose öffentliche Projekte zu hosten, ist GitHub bei Open-Source-Entwicklern sehr beliebt. Entsprechend veröffentlichen ganze Organisationen Ihren Softwarecode auf dieser Plattform und es lohnt sich dieses Netzwerk genauer zu Untersuchen. Aufgrund fehlender Zeit und Rechenleistung Terrabyte von Daten zu untersuchen, wird eine kleinere Organisation genauer analysiert. Hierfür eignet sich die GitHub Organisation NASA sehr gut, weil diese eine überschaubare Grösse besitzt. Die Organisation bietet mit 248 freigegebenen Repositorien und 40 freiwilligen Entwickler (stand September 2019) genügend Daten für eine spannende Untersuchung, die einen Einblick gibt, auf welche Programmiersprachen die NASA setzt und wie die Entwickler aufgestellt sind.

Zu Beginn wird ein GitHub Authentifizierungstoken benötigt, welcher direkt über den Onlinedienst auf dem eigenen Benutzerprofil erstellt werden kann. Mithilfe der Python Bibliothek PyGithub[3] kann eine Verbindung zur GitHub Datenschnittstelle aufgebaut werden. Über diese Datenschnittstelle können Benutzerdaten und sämtliche Daten bezüglich Repositorien frei geladen werden. Das Snippet 1 zeigt auf wie über die Bibliothek PyGithub eine Verbindung aufgebaut wird und im zweiten Teil über einen GitHub-Benutzer sämtliche Repositorien geladen werden. In diesem Beispiel handelt es sich um den Benutzer «nasa».
Mit einer zweiten Python Bibliothek NetworkX[4] werden die geladenen Daten geordnet, sodass diese später über ein Netzwerkanalyseprogramm visualisiert werden können. Snippet 2 zeigt auf wie die gesammelten Daten in eine passende Datei exportiert werden, damit anschliessend mit einem Visualisierungsprogramm einen Netzwerkgraph erstellen werden kann (Matthew A. Russel, 2019).

```python
from github import Github
import networkx as nx

ACCESS_TOKEN = ''
USER = 'nasa'

client = Github(ACCESS_TOKEN, per_page=100)
user = client.get_user(USER)

graph = nx.DiGraph()

for repo in user.get_repos():
    graph.add_node(repo.name + '(repo)', type='repo')
    for lang, number_of_bytes in repo.get_languages().items():
        graph.add_node(repo.name + '(repo)', type='repo', weight=number_of_bytes)
        graph.add_node(lang + '(lang)', type='lang', weight=number_of_bytes)
        graph.add_edge(repo.name + '(repo)',lang + '(lang)', weight=number_of_bytes, type='programming')
```

Snippet 1: Datensammlung für Repositorien und Programmiersprache

[1] GitHub Inc. (2016). GitHub. Abgerufen am 26. 09 2019 von https://github.com
[2] Software Freedom Conservancy. (2019). Git. Abgerufen am 26.09.2019 von https://git-scm.com
[3] Jacques, V. (2019). PyGithub. Abgerufen am 24. 09 2019 von https://pygithub.readthedocs.io
[4] NetworkX Developers. (2019). NetworkX. Abgerufen am 24. 09 2019 von https://networkx.github.io

```
nx.write_gexf(graph, 'datenfile.gexf')
```

Snippet 2: Datenexport für ein Visualisierungsprogramm

Die geladenen Daten sind aufgeteilt in Repositorien und Programmiersprachen. Jedes Repository, dass eine identifizierbare Programmiersprache verwendet, wird mit dieser mittels einer gewichteten und gerichteten Kante verbunden. Die Verbindung wird anhand der Menge an Softwarecode in Bytes gewichtet.
Abbildung 1 stellt den Netzwerkgraph aus den gesammelten Daten dar. Die grünen Knoten repräsentieren Programmiersprachen und die blauen Knoten sind Repositorien. Je grösser ein grüner Knoten erscheint, umso mehr Repositorien verwenden diese spezifische Programmiersprache. Je dicker und dunkler die Kante zwischen zwei Knoten erscheint, desto mehr Programmcode ist in der jeweiligen Programmiersprache im Repository vorhanden. Wie aus Abbildung 1 ersichtlich ist, gehört Python, Shell, Makefile, C, HTML, JavaScript und CSS zu den meist verbreiteten Programmiersprachen in der Organisation. Der Durchschnitt des Netzwerkgraphen liegt ungefähr bei drei Programmiersprachen pro Repository. Dies lässt sich dadurch erklären, dass viele Projekte ein Build-Tool benötigen und ein grafische Benutzeroberfläche bieten, was wiederum von GitHub als eigene Programmiersprache interpretiert wird. Dies wird über den Knoten «Makefile» bestätigt. Bei den meisten Projekten handelt es sich um Schnittstellen, die Daten der NASA Drittpersonen zur Verfügung stellen, oder um Simulationsprogrammen für Hardwarekomponenten. Dies spricht für die Programmiersprachen Python, JavaScript, C und C++, da diese für Schnittstellen Anwendungen und Hardwarekomponenten am geeignetsten sind. Wie durch die grossen Knoten der entsprechenden Programmiersprachen in Abbildung 1 ersichtlich ist.

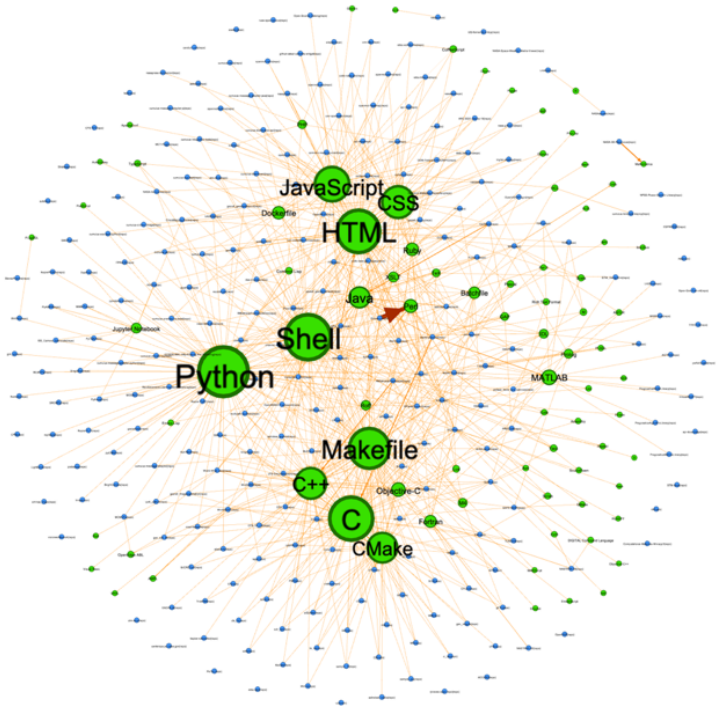

Abbildung 1: Netzwerkgraph Repositorien und Programmiersprachen

In Abbildung 1 sticht in der Mitte eine Kante sehr stark heraus. Hier handelt es sich um die Kante die das Repository «Giovanni» mit dem Knoten «Perl» verbindet. «Giovanni» ist ein Projekt mit

dem geophysikalischen Parameter visualisiert werden können. Beinahe der gesamte Sourcecode ist in Perl geschrieben und benötigt einen Speicherplatz von über 1.6 Gigabyte. Dies wird wiedergespiegelt durch die Dicke der Kante. Es handelt sich hier mit Abstand um das grösste Repository der Organisation, was die anderen Kannten relativ schmal aussehen lässt. Um mehr über solche Repositorien herauszufinden, werden weitere Daten benötigt. Im Snippet 3 werden für jedes Repository sämtliche Commits geladen. Aus den Commits werden die Ersteller ausgelesen. Diese Daten werden erneut mit einem Visualisierungsprogramm dargestellt. Abbildung 2 zeigt den daraus erstellten Netzwerkgraph.

```
for repo in user.get_repos():
    graph.add_node(repo.name + '(repo)', type='repo')
    try:
        commits = [commit for commit in repo.get_commits()]
        graph.add_node(repo.name + '(repo)', type='repo', weight=len(commits))
        users = []
        for commit in commits:
            users.append(commit.commit.author.name)
        users_dict = {user:users.count(user) for user in users}
        for key in users_dict:
            count_of_commits = users_dict[key]
            graph.add_node(key + '(user)', type='user', weight=count_of_commits)
            graph.add_edge(key + '(user)', repo.name + '(repo)', weight=count_of_commits, type='commits')
    except Exception as e:
        print(e)
```

Snippet 3: Datensammlung für Repositorien und Benutzer

Der Netzwerkgraph in Abbildung 2 zeigt auf wie die Benutzerknoten und Repositoryknoten zusammenhängen. Blaue Knoten sind Repositorien und violette Knoten repräsentieren die Benutzer. Die Kante sagt aus, welcher Benutzer in welches Repository committet. Je dunkler und dicker eine Kante ausfällt desto mehr Commits wurde von dem jeweiligen Benutzer erstellt. Die Grösse eines Knoten gibt Auskunft über die Anzahl der Commits. Je grösser die Darstellung, umso mehr Commits sind vorhanden. Aus dem Netzwerkgraph in Abbildung 2 ist zu entnehmen, dass viele Benutzer an einem Repository arbeiten. Sie haben somit eine Aufgabe in der GitHub Organisation NASA. Die Organisation hat aktuell 40 Mitglieder[5]. Jedoch sind im Netzwerkgraph in Abbildung 2 sehr viel mehr Benutzer enthalten. Dies liegen daran, dass nur aktive Mitglieder in der Organisation sind. Sobald ein Benutzer keine Zeit oder Lust hat ein Projekt zu unterhalten, wird dieser aus der Organisation entfernt. Da nun fast alle Benutzer gleich grosse Knoten aufweisen, kann dies bedeuten, dass die Organisation eine grosse «Fluktuationsrate» hat. Dies wird weitgehend über die Internetseite der Organisation bestätigt, da diese immer neue Entwickler für ihre Open-Source Projekte suchen (NASA, 2019).
Weiter ist auffallend, dass die meisten Repositorien von Einzelpersonen gewartet werden. Da die Knoten sehr kleine Punkte darstellen und die Kante sehr hell und dünn sind, bedeutet dies, dass die Benutzer nicht viel an den Repositorien commiten. Das liegt daran, dass es sich bei den Repositorien um fertig entwickelte Projekte handelt, die über GitHub zur freien Verfügung gestellt werden und nicht mehr weiterentwickelt werden. Repository «Giovanni» wurde nur zu Veröffentlichungszwecken zur GitHub Organisation NASA geholt, es wird nicht mehr weiterentwickelt (NASA, 2019).

[5] GitHub Inc. (2019). GitHub. Abgerufen am 30. 09 2019 von https://github.com/nasa

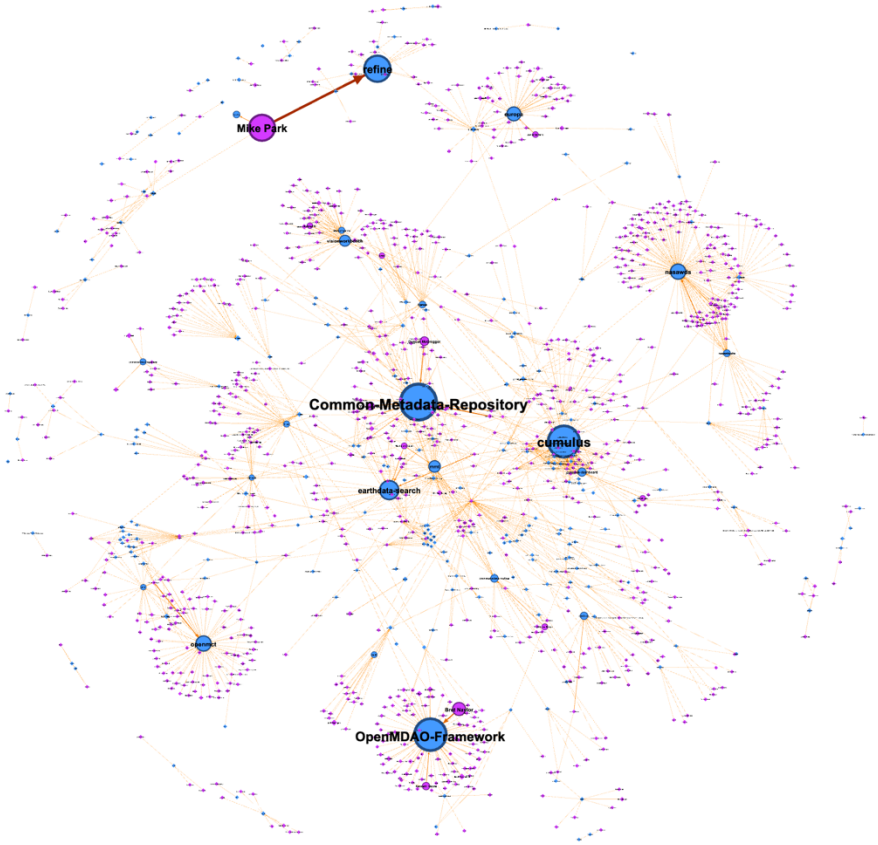

Abbildung 2: Netzwerkgraph Benutzer und Repositorien

Die Untersuchung einer Organisation auf GitHub ist sehr spannend. Natürlich könnte die Organisation noch genauer und tiefer untersucht werden. Beispielsweise könnte untersucht werden, wie beliebt die Projekte sind, oder ob es einen Zusammenhang bezüglich der Stabilität der Projekte und den Forks der jeweiligen Projekte gibt. Jedoch würde dies den Rahmen dieser Arbeit überschreiten. Es ist sehr interessant zu sehen, dass die NASA auf gewisse Programmiersprachen und Build-Tools für die Hardware nahen Programmiersprachen setzt. Die Analysen von Daten mit Netzwerkgraphen ist sehr mächtig. Selbst aus kleinen Datensammlungen, können viele einfache Fragestellungen beantwortet werde. Mit einer grösseren und detaillierteren Datensammlung können mittels dieser Technik sehr genaue Aussagen beziehungsweise auch Voraussagungen gemacht werden. Zudem können Informationen über Organisationen in Erfahrung gebracht werden, die diese vielleicht gar nicht der Öffentlichkeit Preis geben will wie Beispielsweise eine hohe Fluktuationsrate.
Die Analyse kann beliebig auf andere Organisationen oder Benutzer angewendet werden.

Literaturverzeichnis

Matthew A. Russel, M. K. (2019). *Mining the Social Web.* O'Reilly.

NASA. (2019). *Open Government at NASA*. Abgerufen am 29. 09 2019 von https://www.nasa.gov/open

Abbildungsverzeichnis

Abbildung 1: Netzwerkgraph Repositorien und Programmiersprachen 2
Abbildung 2: Netzwerkgraph Benutzer und Repositorien 4

Snippetsversichnis

Snippet 1: Datensammlung für Repositorien und Programmiersprache 1
Snippet 2: Datenexport für ein Visualisierungsprogramm 2
Snippet 3: Datensammlung für Repositorien und Benutzer 3