SemanGit: A Linked Dataset from git

Dennis Oliver Kubitza[1,2], Matthias Böckmann[1,2], and Damien Graux[2,3](✉)

[1] University of Bonn, Bonn, Germany
dennis.oliver.kubitza@iais.fraunhofer.de
[2] Enterprise Information Systems, Fraunhofer IAIS, Sankt Augustin, Germany
{matthias.boeckmann,damien.graux}@iais.fraunhofer.de
[3] ADAPT Centre, Trinity College of Dublin, Dublin, Ireland

Abstract. The growing interest in free and open-source software which occurred over the last decades has accelerated the usage of versioning systems to help developers collaborating together in the same projects. As a consequence, specific tools such as `git` and specialized open-source on-line platforms gained importance. In this study, we introduce and share SemanGit which provides a resource at the crossroads of both Semantic Web and `git` web-based version control systems. SemanGit is actually the first collection of linked data extracted from `GitHub` based on a `git` ontology we designed and extended to include specific `GitHub` features. In this article, we present the dataset, describe the extraction process according to the ontology, show some promising analyses of the data and outline how SemanGit could be linked with external datasets or enriched with new sources to allow for more complex analyses.

Resource type: Dataset
Website: http://www.semangit.de/
Permanent URL: https://doi.org/10.5281/zenodo.2176047

1 Introduction

The semantic `git` (SemanGit), is a novel resource description framework dataset comprising information generated by the `git` protocol and its protocol extensions. So far, it contains nearly 20×10^9 triples about either native `git` protocol data or about social interactions on the `GitHub` platform. While `GitHub` is currently the only used data-source, we designed the underlying ontology to be easily extensible to other providers, such as `SourceForge` [4], `GitLab` [3] and `Bitbucket` [1].

In itself, `git` is a protocol for tracking file changes, such as insertions, deletions or alterations of lines of code, additions or deletions of entire files, etc. The largest online provider for remote `git` repositories is `GitHub` [2]. Besides providing `git` repositories, `GitHub` also implements several social features that are not part of the `git` protocol, such as following other users, watching project changes, creating release versions of a project and pull requests – a request for the contributors of a repository to adapt provided source code changes.

Our goal was to add semantics to the `git` protocol to make use of the strengths of both semantics and graph databases in general, such as interlinkage with other datasets or general graph traversal tasks. Particularly, by using RDF, as recommended by the W3C, we ensure an easy integration with other datasets from the Linked Open Data cloud[1]. The backbone and initial step of our project was the design of an ontology to support the logical inference.

Due to the vast size of `GitHub`, we chose to extract our data from this provider for creating our resource description framework, our first data source. They provide a REST API [11] as a query point from which one can gather data. With its limitation of 5,000 queries per hour per token, it would take over 2 years to query all 100 million repositories [10] just once, yielding only a fraction of the data.

The `GHTorrent` project [12] however provides large amounts of data which they have gathered from `GitHub` using a multitude of tokens over several years, offering us a better input than the rate limited `GitHub` API. Its data is stored in a relational model and therefore not well suited for analysis of linked data, where a graph dataset is preferred instead. To bypass this issue, we wrote a Converter transforming the relational tables into a `RDF`. Since our input data is already several hundred gigabytes in size, we were forced to optimize our output as much as possible, while still ensuring valid Turtle syntax [8].

The datasets are subject to the *CC BY-SA 4.0* license (see [9] for more details) and are available under:

www.semangit.de

The latest version of our implementation is available under the following link:

https://doi.org/10.5281/zenodo.2176047

The rest of this article is structured as follows. Firstly, we present background knowledge in Sect. 2 to provide the reader with concepts coming from open Open-Source communities. Then, in Sect. 3, we review the related research efforts close to our approach. Next in Sect. 4, we describe the vocabulary and the ontology we particularly designed to generate the dataset. Afterwards, in Sect. 5, we present the shared resource we developed, before presenting some sample analyses in Sect. 6. In Sect. 7, we collect further interesting use cases. Finally we conclude in Sect. 8 and round everything up by elaborating our sustainability plan and future works.

2 Preliminaries

In this Section, we recall some important concepts and standards which will then be used during the detailed description of the SemanGit.

Developed in 2005, `git` [16] is a system aiming at tracking changes in a file system while providing several properties such as data integrity or support for distributed and non-linear workflows. As a consequence, it has been adopted

[1] https://lod-cloud.net/.

by the Open-Source developers as a tool to work concurrently on large shared projects. Since the file system represented by a `git` repository can be distributed, developers embed their changes into a `git` repository and once they are ready "push" their contributions to the "bare"-repository so that collaborators can then have access to the latest version.

Quickly, the `git` protocol has evolved to provide more and more features dedicated to large Open-Source communities and projects. These features comprise for example the possibility of creating new branches for a project where a sub-group of contributors can develop additional features independently, which could thereafter be merged back into the "master" branch.

3 Related Work

To the best of our knowledge, SemanGit is the first open attempt to systematically build a linked dataset from a group of `git` repositories.

Nonetheless, SemanGit is not the first effort that aims towards extracting and grouping information from an open `git` platform. Indeed, Gousios in [12] introduced the `GHTorrent` project which aims at providing data dumps extracted from the `GitHub` public API.

To be precise, the SemanGit project falls within the domain of transforming public data into linked data. So far, numerous projects are already providing such datasets each one tackling a distinct domain. Among this list, we can mention DBpedia [6] which proposes a linked version of Wikipedia, or also [7] which deals with geographical data. For a more exhaustive list, we refer the reader to the Linked Open Data cloud which groups 1,239 distinct datasets as of March 2019.

4 A `git` Dedicated Vocabulary

The `git` protocol in itself relies on so-called repositories in which data can be stored. Many online `git` repository providers add some features of their own that are not part of the `git` protocol, such as social features. In order to have an extensible ontology, we need to clearly distinguish between what is part of the `git` protocol and what is provider specific. As an example, according to the `git` protocol, the author of a commit is simply a pair "Name <email>" whereas on `GitHub` an author, i.e. a user, is much more complex. It has additional attributes such as a creation date, an avatar, a location and even social-featured ones such as an associated website.

The part of the ontology covering the `git` protocol features only represents the data that strictly belong to the protocol. The classes in this section mostly form the basis from which platform-specific classes inherit, see Fig. 1 for an example.

This protocol-related part is rather small and comprises of merely four classes: Users, projects (i.e. repositories), commits and pull requests, the user class storing no more than an email address. The projects refer to a URL, a timestamp of creation and the commits that were submitted to it. The other

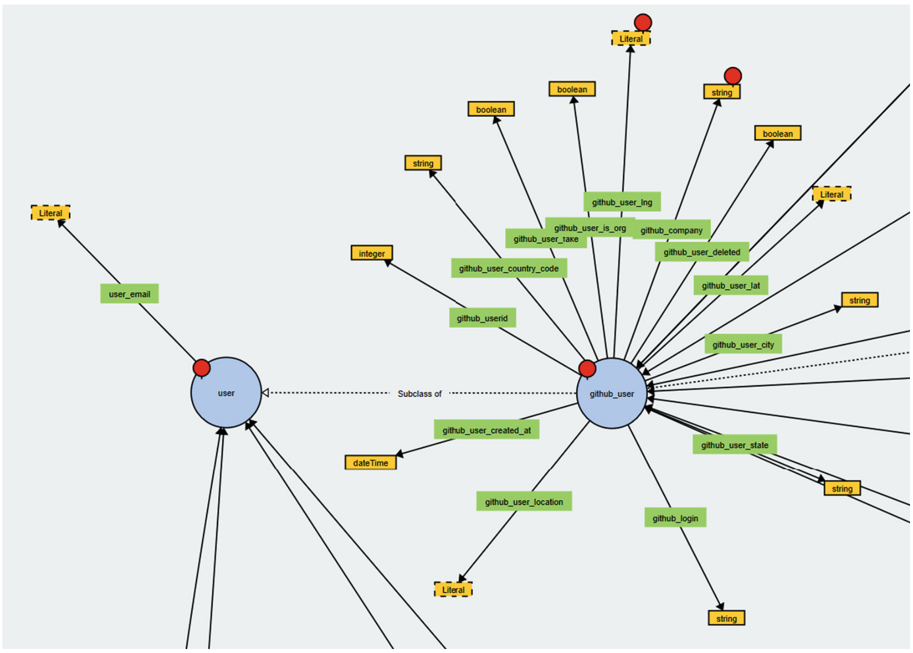


Fig. 1. A visualization with WebVOWL [15] of a small section of the ontology

two classes are slightly more complex, as commits have a hierarchical structure in themselves and pull requests are requests to accept a cross-project commit.

Seeing that all extensions of the `git` protocol are still required to provide the base functionality, we have chosen a hierarchical approach for our ontology, letting extensions inherit from protocol-conform classes to make them take over all properties they are required to have.

These classes corresponding to provider-specific extensions of the protocol are set apart from the original one by putting a prefix such as "github_" to the class name. Large parts of our ontology do not refer to parts of the `git` protocol but try to encompass those features that have been added or extended by providers on top of it. Some of them are purely social relations, such as one user following another, or multiple users forming an organization. Others are actual versioning features, like forking of projects and issue tracking. `GitHub` allows users to leave comments on certain objects, such as commits and pull requests, which is not specified in the `git` protocol, which only allows for an initial commit message. In such a case where an entire feature has been added that is not an extension of an existing `git` protocol feature, the corresponding class in the ontology does not inherit from a class that represents a `git` feature. An example of this is the issue tracking system implemented by `GitHub`, see Fig. 2. The full ontology file can be found on `GitHub`[2] and an interactive visualization can be found on `VisualDataWeb`[3].

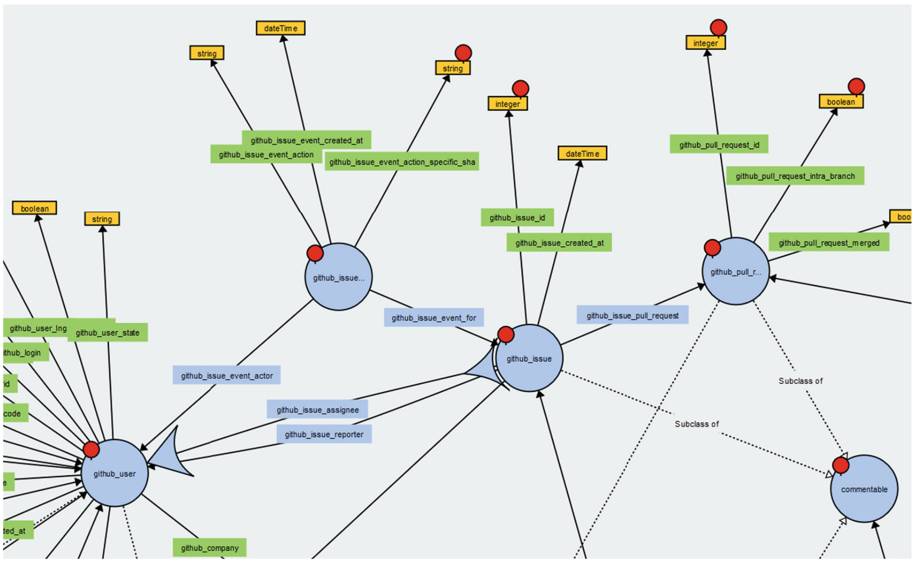


Fig. 2. A visualization with WebVOWL [15] of a provider specific feature

5 Creating the SemanGit Dataset

We will start off this chapter by giving detailed information on how the SemanGit dataset is created. Afterwards, we present statistics on the effectiveness of the steps we have taken to reduce the disk size of the output.

5.1 Data Generation Process

One could now take the most direct path and start querying `GitHub` via their REST API [11]. This approach faces the drawback of running into limitations regarding the number of queries one can fire at the API per hour, which is currently at 5,000 queries per token per hour. Hence, even with the drastic underestimation of only requiring one query per repository to get all relevant information, it would already require multiple years to query all repositories just once, of which there are more than 100×10^6 as of November 2018 [10]. The `GHTorrent` project [12] has been mining meta information from `GitHub`

[2] https://github.com/SemanGit/SemanGit/blob/master/Documentation/ontology/semangitontology.ttl.

[3] http://visualdataweb.de/webvowl/#opts=doc=0;editorMode=true;#iri=https://raw.githubusercontent.com/SemanGit/SemanGit/master/Documentation/ontology/semangitontology.ttl.

since 2013, using several access tokens in parallel. They offer monthly database dumps which we use to get around the query limitations of `GitHub`'s API.

The monthly data dumps are provided in the form of comma separated value (CSV) files which store different objects or certain object relations. As an example, there is one file storing all users and one each for the social interactions of following another user or watching a project for updates. Having different files for different kinds of relations allows for some trivial parallelization, even though this is rather ruined by the fact that two files are larger than all of the rest put together: The files storing the commits and to which project they belong add up over 60% of the total dump size.

Given our ontology and the input from the `GHTorrent` project, the process of writing a translation tool to convert the CSV files into Turtle is quite straightforward. Considering the size of the dataset though, it seems prudent to spend additional effort on compressing the output as much as possible.

The Java converter source code is available in the following `GitHub` repository:

https://github.com/SemanGit/Converter

We will now give some details on the actual conversion process and outline some tricks used to reduce the size of the resulting RDF file.

We have created a bash script to automate the processing by checking for new data dumps, managing the download, decompression and ensuring fault-tolerance for the used resources. For each step, we have added error checks and fallback mechanisms to guarantee the integrity of the result. These checks are mainly log files, documenting which tasks have been completed up to which point so that we can restart the process at a suitable point. It is for example not required to re-download the dump if the machine runs out of space while extracting, or to re-extract if an issue is encountered during the conversion.

To keep the size of the output as small as possible, we have made the obvious choice of serializing our data in the Turtle format [8], giving us the ability to use prefixes and to abbreviate parts of triples. Seeing that we are working on a fixed ontology, we have taken it to the extreme of creating one prefix for every URI in our ontology, choosing prefix names no longer than two characters and choosing the shortest ones for the most commonly used URIs, such as the empty prefix for the repository resource, which occurred more than 7.7×10^9 times as subject or object.

```
@prefix semangit: <http://semangit.de/ontology/semangit>.

# Unoptimized Data
semangit:ghissue_123456 a semangit:github_issue;
semangit:github_issue_created_at "2002-05-30T09:00:00"^^xsd:dateTime;
semangit:github_issue_project semangit:ghrepo_234567;
semangit:github_issue_assignee semangit:ghuser_345678.

# With prefixing
u:123456 a x:;
```

```
C: "2002-05-30T09:00:00"^^xsd:dateTime;
y: :234567;
A: m:345678.

# With Base64 like integer representation
u:x3T a x:;
C: "2002-05-30T09:00:00"^^xsd:dateTime;
y: :WR9;
A: m:af93.
```

Additionally, the data from `GHTorrent` is presorted, coming from relational tables, maximizing the number of abbreviations possible. Lastly, we were able to reduce the output size drastically by transforming all integers in resource identifiers from the base 10 representation to a base 64 like representation, that is compatible with Turtle syntax.

After the data generation process is finished, we describe the resulting dataset with the Vocabulary of Interlinked Datasets (VoID) [5]. These triples include a name and description for the dataset, its format, the license under which it is available, links to the associated homepage, modification and creation date, author contact details and more.

5.2 Statistics on the Dataset

At the time of writing this study, the most recent version of SemanGit is from April 2019 and has a size of 353 GB with over 21 billion triples. The input files from GHTorrent use 340 GB of space, which means we create less than 4% overhead by adding semantics, which is owed to the measures we described in the last section. By using prefixing to the extent that the turtle format allows, we achieved to be little more than 25% larger than the input files. By also adding a Base64 like integer representation, this overhead was reduced to the above mentioned 4%. The entire conversion process was completed in less than seven hours.[4] Our dataset contains 31,205,000 users, which is slightly more than the number of users GitHub claims to have had in November 2018 [10].

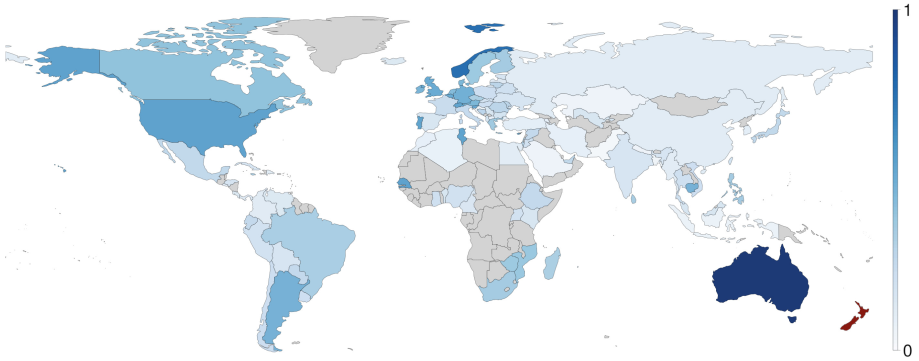
6 Example Analyses

To emphasize the value of the dataset and its structure, we have created two sample analyses, computed on our previously mentioned server. In the first one, we take a brief look at international cooperation of developers for the countries New Zealand and Germany. In the second analysis, we compute statistics for the internal structure of organizations, by using the `follow` relation. To present the potential for knowledge discovery, we extract the data of some colleagues, who work in different roles for the same organization and compare them to these statistics.

[4] Intel Core i7-5820 CPU @ 6×3.3 GHz, 64 GB DDR3, Ubuntu 18.04.

6.1 Global Cooperation Within Repositories

Many users on `GitHub` state the country they live in or originate from. With this data, we can derive interesting analysis for countries on a global scale, enabling comparison on regional differences for programming languages, coding style, social media behaviour or even business policies. Besides comparing differences regarding those aspects, one can also analyze how well countries cooperate. A repository can have multiple collaborators. The SemanGit dataset represent this information through the class `project_join_event`, linked to a user and a project. For the sake of demonstration, we will analyze which nations New Zealanders collaborate with frequently. In two separate queries, we collect for all countries the absolute number of users $N_{country}$ and the number of users working together with at least one New Zealander in a project $N_{country,NZ}$. The total execution time for both queries was below 9 m. As our triplestore makes use of intermediate results, it is difficult to measure the runtimes of queries independently, without resetting the server after each query.



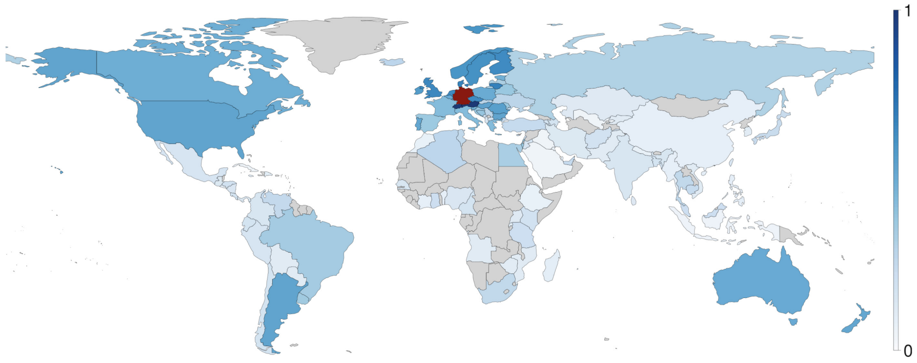
Cooperation Index $I_{country,NZ}$ represented in shades of blue
Excluded countries in gray. New Zealand in red.

Fig. 3. Cooperation with New Zealanders (Color figure online)

From these data, we calculated the share of people per country collaborating with New Zealanders $S_{country,NZ} = \frac{N_{country}}{N_{country,NZ}}$ and normalized the results to form an index.

$$I_{country,NZ} = \frac{S_{country,NZ}}{\max_{c \in Countries}(S_{c,NZ})}$$

Figure 3 contains the results, showing a strong collaboration with New Zealand's neighbour Australia (1.00), but also with countries like Norway (0.75), Senegal (0.54), the United States (0.54), Switzerland (0.53), Portugal (0.53), Tunisia (0.53), Slovenia (0.51) and Cyprus (0.51). For comparison, we applied the same procedure for Germany, revealing that collaboration with other German-speaking countries is more common than with other nations, see Fig. 4.



Cooperation Index $I_{country,DE}$ represented in shades of blue
Excluded countries in gray. Germany in red.

Fig. 4. Cooperation with Germans (Color figure online)

6.2 Social Relations for Organizations

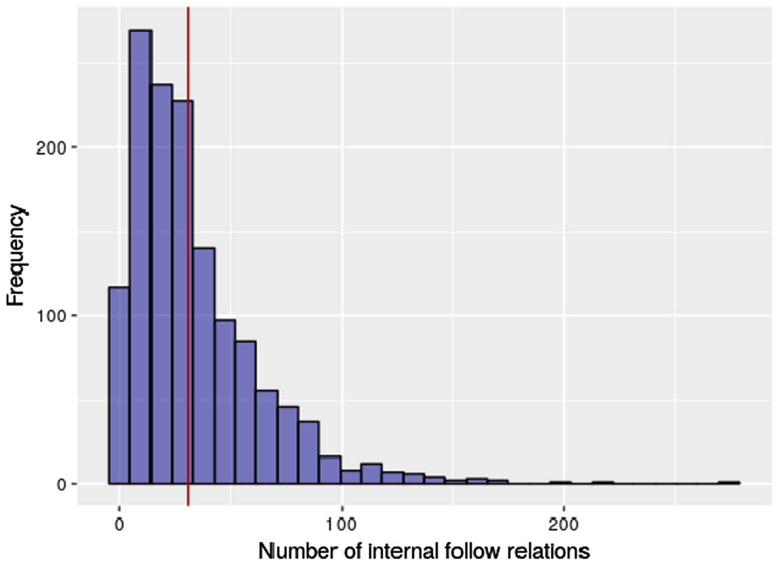
To glimpse at possible investigations of social aspects in Open Source collaboration, we analyzed two features: the number of members of an organization and the number of followers a user has. In the SemanGit dataset the class user corresponds either to natural user, or an organization. In the second case, the organization has different members, which are also users. We track the membership with `organization_join_events`. By agglomerating over all these events, it is possible to get the set of all members. Also SemanGit tracks which user follows whom on `GitHub` with the `follow_event`, also linked to two users.

We have the hypothesis, that we can learn something about the internal structure of an organization by looking at the behaviour of their users. Without information about the real structure of these organizations this kind of analysis would fall into the category of unsupervised learning. To avoid the application of machine learning, we investigated an organization for which we know the internal structure and roles of people. This anonymous organization `Org` comprises 19 members and overall 31 follow relations. To obtain an overview about the dataset, we queried all organizations, the cardinality of their members and internal follow events. The Listing 1.1 provides a precise description of how, practically, the results are extracted from SemanGit. Indeed, the query is used to return a sorted list (see line 18) of triplets (lines 2 to 4); the `optional` section is used to collect the internal following relationships.

```
PREFIX sgo: <http://semangit.de/ontology/> .
SELECT ?organization
       (COUNT(DISTINCT ?user1) AS ?users)
       (COUNT(DISTINCT ?follow_event) AS ?follows)
WHERE
{
  ?organization sgo:github_user_is_org true .
  ?join_event_1 sgo:github_organization_is_joined ?organization ;
                sgo:github_organization_joined_by ?user1 .
  # Collect the internal follow relations
  OPTIONAL
  {
    ?join_event_2 sgo:github_organization_is_joined ?organization ;
                  sgo:github_organization_joined_by ?user2 .
    ?follow_event sgo::github_follows  ?user1 ;
                  sgo::github_follower ?user2 .
  }
} GROUP BY ?organization}
```

Listing 1.1. SPARQL query used to extract the needed information from organizations

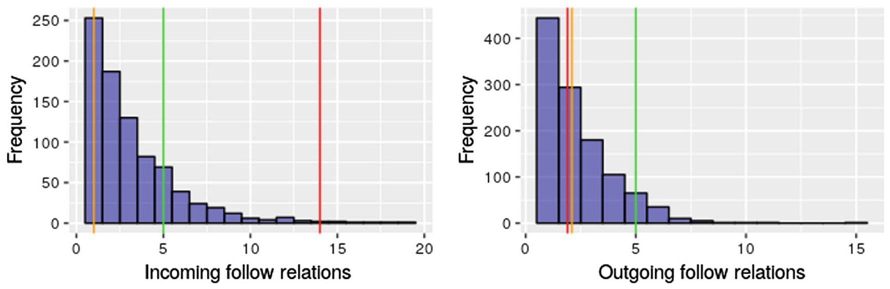
The resulting data contains almost $300,000$ organizations. From this data we selected all $1,375$ organizations with 17 to 22 members and printed the distribution of internal follow relations compared to `Org`. Figure 5 shows these results, with `Org` being located slightly after the peak, showing it is neither significant low nor high numbers of follow relations.



Histogram for organizations with 17 to 22 members.
Observation for a well known organization as vertical-line on $x = 31$.

Fig. 5. Number of follow relations within organizations

An investigation against organizations with a significantly different number of such relations, would be meaningless. Therefore, we picked the 93 organizations with 17 to 23 internal relations to compare against `Org`. For each user affiliated to one of these organizations we now queried for two informations: the number of followers, and the number of people she follows. The outcomes are plotted in Fig. 6. For `Org` we observed two users with 14 and 5 followers, already summing up to 55% of the follow relations. Querying their names revealed, that these are the developer responsible for maintaining the repositories on `GitHub` and the department leader. Overall, six of the colleagues were not followed at all, five of them having assisting positions.



Histogram for organizations with 17 to 22 members and 29 to 33 follow relations. Observation for different roles within the well known organization as vertical lines. Leader of Research Group (green), Post Doctoral Researcher and Main Developer (red), PhD Student (orange)

Fig. 6. Followed within an organization (Color figure online)

We do not claim that our hypothesis is true according to these results, but they provide at least a hindsight about the value of the contained information.

7 Further Use Cases

The sample analyses from Sect. 6 already indicate that the domain of use cases is quite diverse. We will now present a few sample use cases to demonstrate the potential value of the dataset.

The Headhuntress – Finding computer scientists to hire can be quite challenging. Suppose a headhuntress is looking for quality programmers to employ. While it is easy to determine how high-quality programming is reflected on GitHub (no major issues reported, positive comments on projects, no infrequent commits), it is more problematic to actually measure and compare these traits. With SemanGit, she can find a representative subpopulation and generate benchmark results either by doing analysis manually or applying machine-learning. Furthermore, the dataset contains geographical information about many users, which is geocoded from the *location* field on a users profile. With the DBpedia interlinkage [6], one could, for example, look for users inside or close to a given city by using the *nearestCity* relation. She can even attempt to find well socialised, skilled programmers by adopting an analysis as mentioned in Sect. 6.

```
PREFIX sgo: <http://semangit.de/ontology/> .
SELECT  ?other_project
        (COUNT(DISTINCT ?user) AS ?users)
WHERE
{
  ?project sgo:github_repo_name "SemanGit" .
  ?project_watch_event1 sgo:project_is_followed ?project ;
                        sgo:github_project_followed_by ?user .
  ?project_watch_event2 sgo:github_project_followed_by ?user ;
                        sgo:github_project_followed ?other_project .
} GROUP BY ?other_project
  ORDER BY DESC(?users)
```

Listing 1.2. SPARQL query used in the context of the *developer* use case

The Developer – The records of social interactions on GitHub in our dataset can be used for more than just social analysis. Assume a developer has been working with an open-source tool and is looking for an alternative tool. With SemanGit, he could try to find similar projects by taking the set of developers who are watching the tool's repository and evaluating the set of repositories these developers are watching (see e.g. Listing 1.2).

The Economist – One topic of economics is the analysis of driving forces, structures and institutions of an economy. While the behavior of agents in traditional scenarios are well-documented, the analysis of Open-Software-Projects and the motivation behind contribution is subject to notable current research [13]. For such research interests, the use of linked data offers new opportunities as it is tailored for the analysis of local models and offers direct access to empirical data on individual level.

8 Conclusions, Future Work and Sustainability

In this article, we presented and shared the SemanGit dataset which is a linked data version of `GitHub` activities. It already consists of more than 20 billions RDF triples. In addition to the openly available dataset, we also provided the extractor in our GitHub repository, which converts the data from GHTorrent to ontology compliant RDF, and the ontology we designed to represent `git` repositories and `GitHub` activities.

As explained, the SemanGit structure is prone to be extended by considering adding new "social feature" related terms to the already existing ontology in order to include other `git` platforms such as `GitLab` for instance. In addition to this horizontal extension, we are already orienting our next efforts towards the computation of several layers of analysis as presented above. Moreover, in order to offer an even more complete dataset, we are currently exploring directions to link even more the dataset with already well-established data such as e.g., DBpedia [6] or DBLP [14]. SemanGit is currently still lacking a public SparQL endpoint, making the full dataset available. This task is rather challenging due to the size of the data. Currently we commit our computational resources to the inclusion of the vertical extensions and linking with other datasets. After the necessary computational power and storage will be freed, we plan to implement an endpoint on our server.

More generally, even if the regular extraction process is still recent, the SemanGit dataset already has several directions of development. It will be sustained by several European projects on which we are contributing right now e.g. the QualiChain project. As a consequence, we will maintain the project at least until 2022, by providing the most recent datasets in bi-monthly intervals and continuing the development. To ensure that all dumps are recreatable, even if not listed on our homepage, the extraction and converter tools remain in the public repositories on `Github`.

We have built the SemanGit dataset having in mind a large number of possibilities it would offer, thus we do hope it will soon be considered as a bridge between the Open-Source and Semantic Web communities.

Acknowledgment. This work is partly supported by the German Federal Ministry of Education and Research (BMBF) in the context of the research project “Industrial Data Space Plus” (GA 01IS17031) as well as the Fraunhofer Cluster of Excellence “Cognitive Internet Technologies” (CCIT); by the EU H2020 project “QualiChain” (GA 822404); and by the ADAPT Centre for Digital Content Technology funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded under the European Regional Development Fund.

References

1. Bitbucket. https://bitbucket.org/. Accessed 16 Aug 2019
2. Comparison of source code hosting facilities. https://en.wikipedia.org/wiki/Comparison_of_source_code_hosting_facilities. Accessed 16 Aug 2019
3. GitLab. https://about.gitlab.com/. Accessed 16 Aug 2019
4. SourceForge. https://sourceforge.net/. Accessed 16 Aug 2019
5. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Void guide—using the vocabulary of interlinked datasets. Community Draft, voiD Working Group (2009)
6. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
7. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: adding a spatial dimension to the web of data. In: Bernstein, A., et al. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 731–746. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04930-9_46
8. Eric Prud’hommeaux, G.C.: RDF 1.1 turtle: terse RDF triple language. http://www.w3.org/TR/2014/REC-turtle-20140225/, The latest edition is available at http://www.w3.org/TR/turtle/

9. Attribution-Sharealike 4.0 International (CC BY-SA 4.0). https://creativecommons.org/licenses/by-sa/4.0/. Accessed 16 Aug 2019
10. GitHub About. https://github.com/about. Accessed 16 Aug 2019
11. GitHub REST API. https://developer.github.com/v3/. Accessed 16 Aug 2019
12. Gousios, G.: The GHTorrent dataset and tool suite. In: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR 2013, pp. 233–236. IEEE Press, Piscataway (2013). http://dl.acm.org/citation.cfm?id=2487085.2487132
13. Lerner, J., Tirole, J.: Some simple economics of open source. J. Ind. Econ. **50**(2), 197–234 (2002)
14. Ley, M.: The DBLP computer science bibliography: evolution, research issues, perspectives. In: Laender, A.H.F., Oliveira, A.L. (eds.) SPIRE 2002. LNCS, vol. 2476, pp. 1–10. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45735-6_1
15. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: web-based visualization of ontologies. In: Lambrix, P., et al. (eds.) EKAW 2014. LNCS (LNAI), vol. 8982, pp. 154–158. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17966-7_21
16. Torvalds, L., Hamano, J.: Git: fast version control system (2010). http://git-scm.com