Team 81
Github Exlorer!

Analyzing the Implicit Social Network from GitHub Activities

Ran Tavory **Jonathan** del Campo **Ruzvidzo** Ngulube

Summary

We develop a method of mining GitHub event activity to construct an implicit social network between GitHub users encompassing their GitHub repositories and the connections between them. We then utilize this network for interesting visualizations, including shortest path between users, and finding the most influential users for example in a technology scope.

Businesses may care about community discovery for marketing purposes. Individuals may care about personal branding and achievements.

Approaches

- Collect data from GitHub's Archive website, iteratively process this data by extracting key information on Repositories and Actors/ Users and load that into a Neo4j graph database.
  - Repositories, Users/ Actors are created as Nodes
  - An actor to Repo relationship is established

Data

There are multiple sources on the internet for GitHub Data.

We settled on the following 3 sources
- Google BigQuery (BQ) - for exploratory data analysis
- GitHub's API - for getting additional information about a repository or actor/user.
- GitHub's Archive (GHA) - easier to work with in a batch and more efficient.

We used python to download houlrly archive files from GitHub's Archive website and processed the relevant fields into a neo4j databse. Table 1. below shows counts of the nodes and relationships within our graph database.

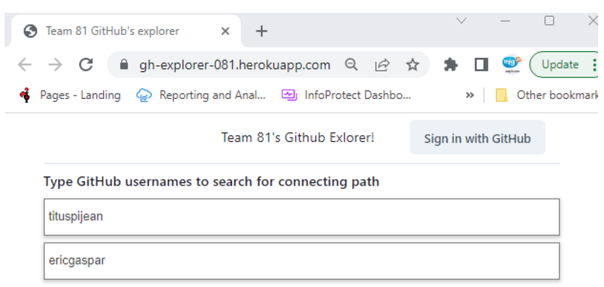
Table 1. Neo4j Database Entity Type and count

Type	Label	Description	Count
Node	Repo1	Repository Node	154 763
Node	Actor1	Actor/ User Node	104 746
Relationship	TO	Actor1 to Repo1 Relationship	132 558

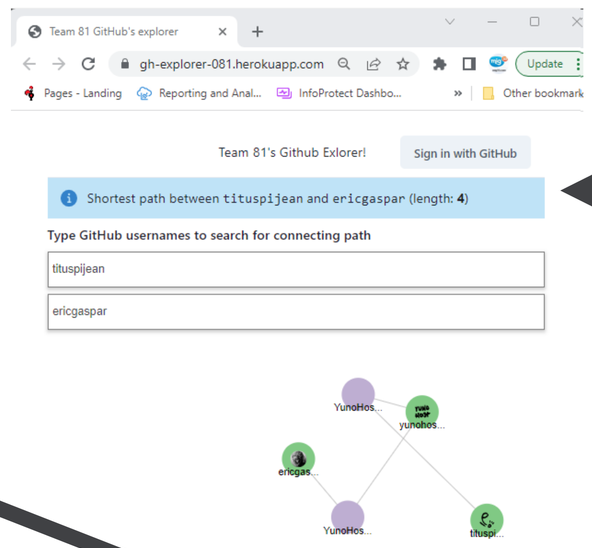
Storage consumed by the above database is approximately 78Mb. The data is temporal especially when one starts to look at interactions over time.

User Interaction Application Approach

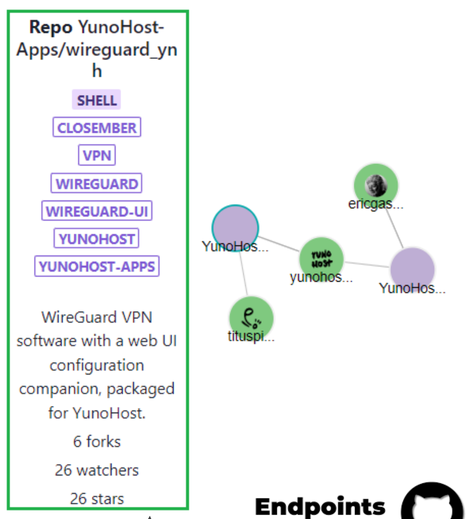
Step 1.
User vistis Team 81's Github Exlorer! Search for two GH users by using the two text boxes to find the shortest path between.e.g.**tituspijean** in top box and **ericgaspar** in the bottom one. With typeahead, auto-completion makes it easy to find users



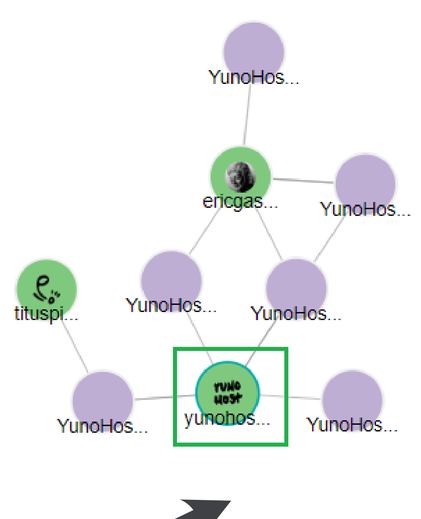
Step 2.
UI queries the Neo4J database and returns shortest path between the 2 users if it exists



Step 3.
Clicking on any node, will display more details about that node. This is retrieved from GitHub on-demand via the GitHub APIs



Step 4.
Double clicking on some nodes will expand ithe node to one level to its neighbours. e.g. double clicking thegreen user yunohost with the bold text



Experiments and Results

We performed the following experiments.
- Usability. How intuitive is it to discover the shortest path between GH users and iteratively expand the nodes in the graph? We found that most users find it very intuitive and easy to use. We tested with co-workers and family members. It is easy to find users with the help of the typeahead textboxes.
- Scalability. The source data (on GHA or BQ) is huge. One concern was whether it is practical to insert it into a reasonably sized Neo4J database. We find through experimentation that it is certainly possible after filtering out a lot of data that we consider noise and distilling only the data we recognize as important.
- Data signal-to-noise ratio. One experiment we did not plan ahead for but realized we have to do is to increase the data's signal-to-noise ratio. We realized that many of the interactions are performed by bots on GitHub (automated actors, not real users). So, staying with our goal of revealing true connections in GH we decided to remove all bot interaction (106 bots found out our data and 100x connected edges)

Technology Stack