Feiyu Chen, feiyuche@mit.edu, f0bdb25e571b8f0f81e77511f25cf8faa5e7861a

Github: The Push Events on 2019-10-01

Our team is working on Github graphs for the final project, so in this lab, I took the chance to have a closer look at the dataset we will be using. The visualizations are included in this writeup.

Dataset

GH Archive (https://www.gharchive.org/) is a public archive of the history of Github. It includes logs of 20+ event types (e.g. push to a repo, fork a repo). The data on each event consists of the user who initiates the action, the repo this action targets, and other information. The data are dumped on an hourly basis and is in the form of JSON files.

Github has a large volume of event data. On average, an hourly JSON dump is ~300MB. Because of limited computing resources, in this lab, we focus on the push events generated on the day 2019-10-01.

Preprocessing

We care about 3 entities in our dataset: users, repos and events. The occurrence of different event types associated with every user is summarized, so is the occurrence of different event types associated with every repo.

Each event is represented by a (user, repo, event type) triplet. We want to visualize the triplets as a graph in which users and repos are nodes and events are edges. For the 2019-10-01, there are ~2000,000 such triplets, which is too many to visualize. If we only keep the push events, we get ~1000,000 such triplets, which is still too much. If we plot a histogram of the push events associated with each repo (Figure 1), we can see that the majority of the repos receive no pushes or very few pushes. A small portion of the repos receives more than 10 pushes. We chose to focus on these repos for this lab and kept only the push events associated with these repos.

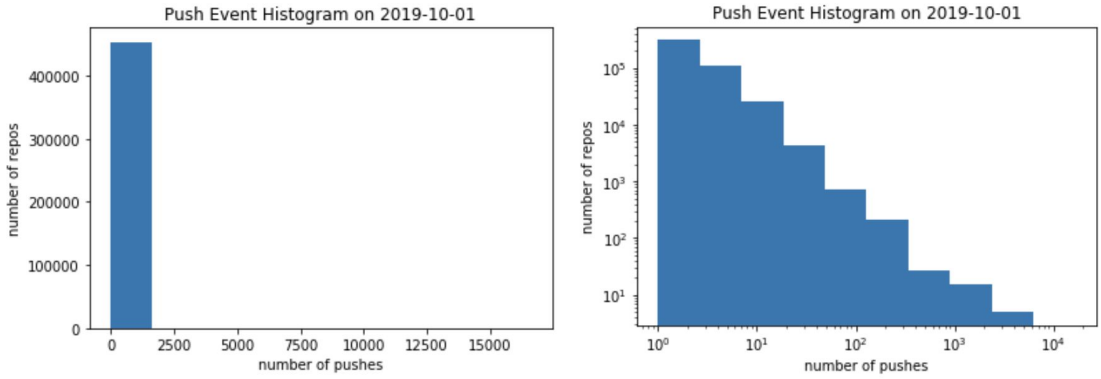

Figure 1.
Histograms of push events associated with each repo. Left: linear scale. Right: log scale

Visualization

Gephi (https://gephi.org), an open-source graph visualization platform written in Java, is used to visualize the graph. The events were written into 2 CSV files, one describes the edges (user as source column, repo as target column), another describes the nodes. The edges and nodes were imported in Gephi. Yifan Hu algorithm is used to adjust the layout of the graph. In the end, the nodes are colored by their type (with the give-color-to-nodes plugin), nodes representing a repo were colored red and nodes representing users were colored blue. The following is the overall graph (Figure 2):

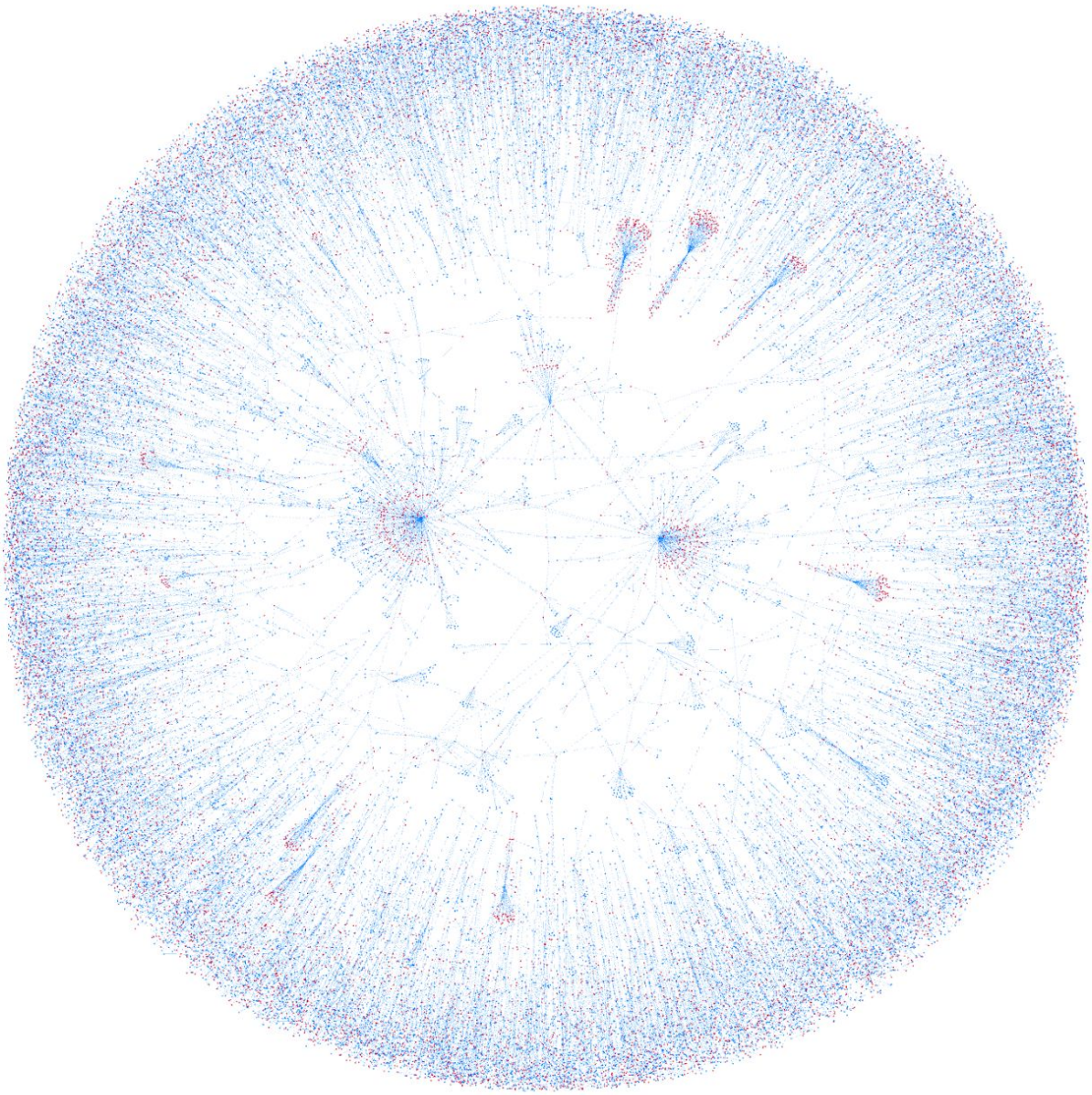

Figure 2.
Graph of push events related to popular repos. Red dots are repos and blue dots are users.

Observation

-The graph is sparse.
The average degree of this graph is ~0.6. If we plot the degree distribution (Figure 3), We can see that almost all nodes have a degree lower than 5. This makes sense because most users don't push to multiple repos in a day. Considering we are already filtering out the repos that receive less than 10 pushes on that day, we can imagine that the original graph is even more sparse.

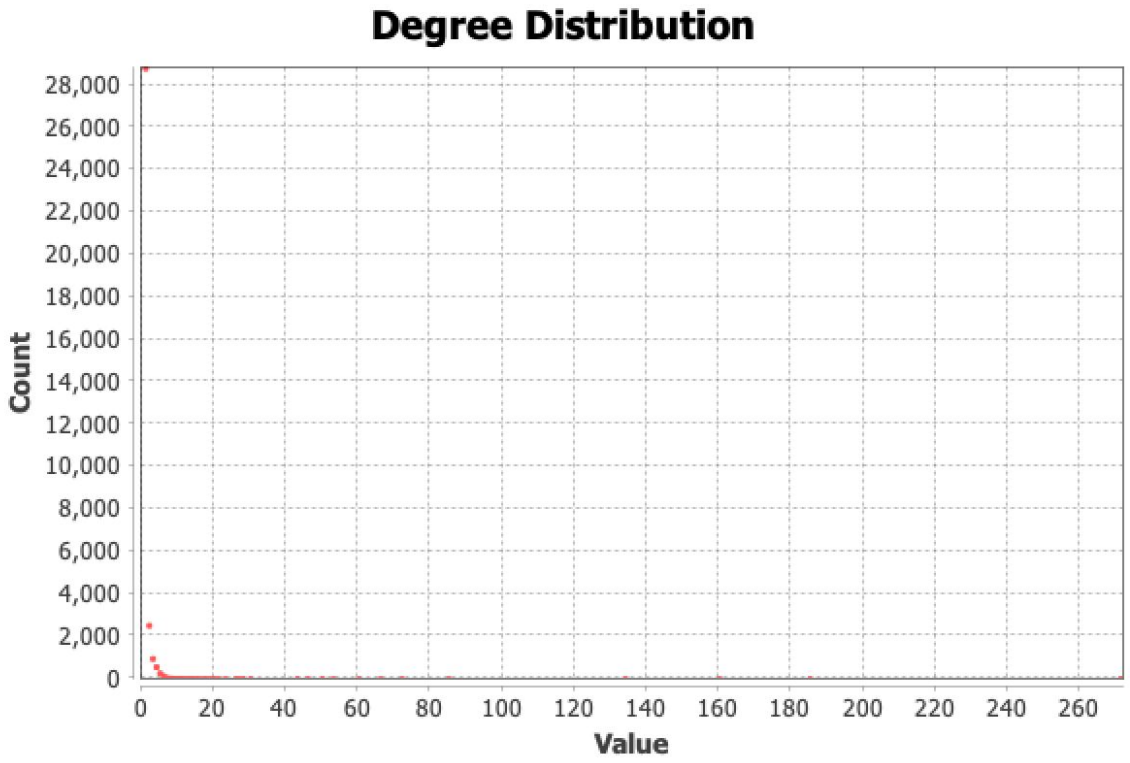

Figure 3.
Degree Distribution

-High-degree repos are usually popular open-source software or temporary repos for pedagogical purposes.

There are some 'clusters' in our graph where we have many nodes points to a single node. Sometimes time the node in the center represents a user, in other cases, it represents a repo. Those mid-sized 'clusters' are mostly centered around a repo (Figure 4). If we look up these repos, they are usually popular open-source software. For example, in Figure 4, we have CoreFX (the foundational class libraries for .NET Core), Elasticsearch (distributed search engine for text-search), Calypso (Wordpress.com frontend). Besides these, we also have some temporary repos set up for a lecture or tutorial.

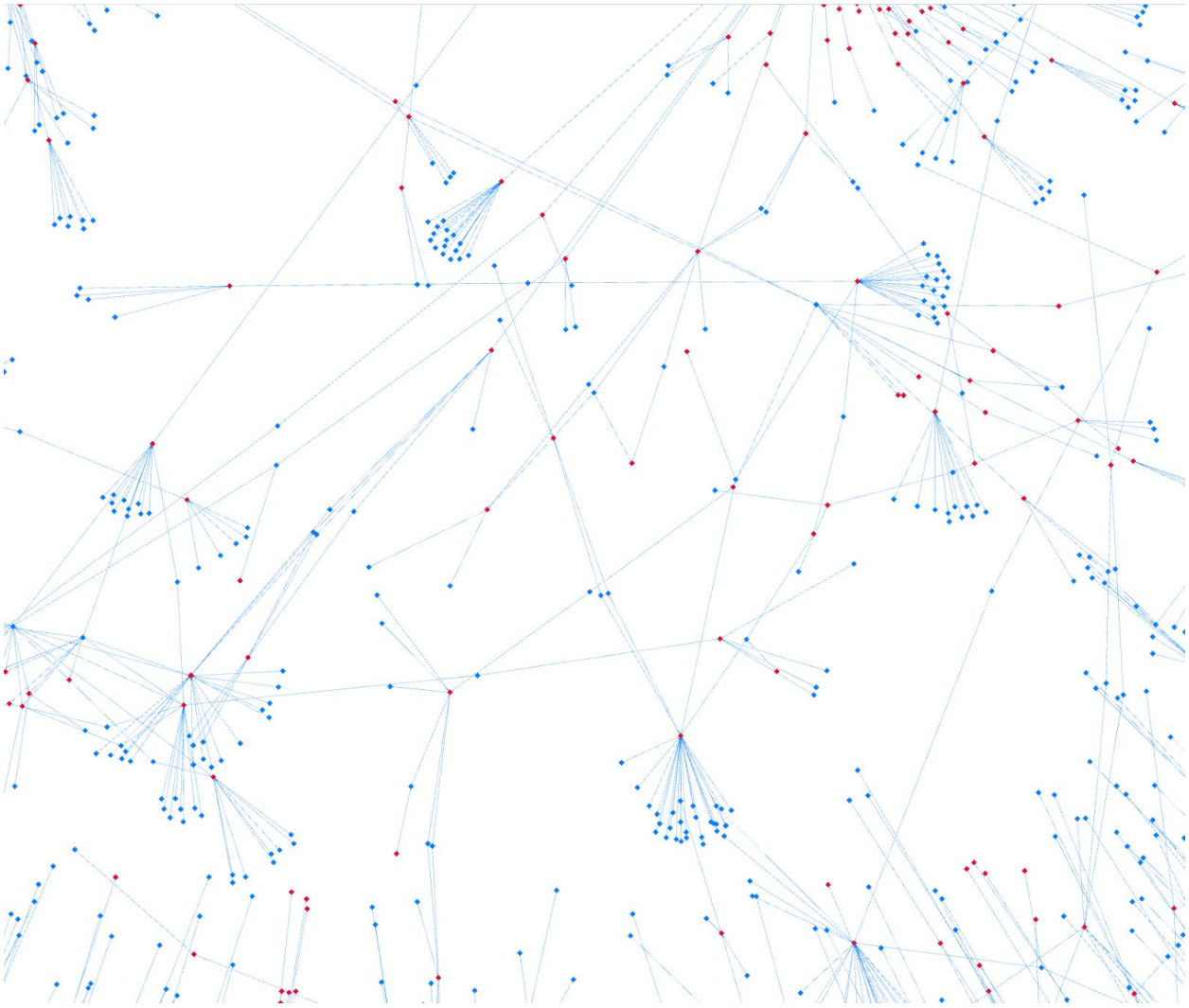

Figure 4.
High-Degree Repos

-High-degree users are usually bots.

There are also ‘clusters’ that centered around a user. These ‘clusters’ tend to be larger. The user in the center is usually not a real person. For example, the 2 largest such ‘clusters’ center around dependency management services (Figure 5): Dependabot and Renovate. These services check for outdated requirements and push to repo to update dependencies.

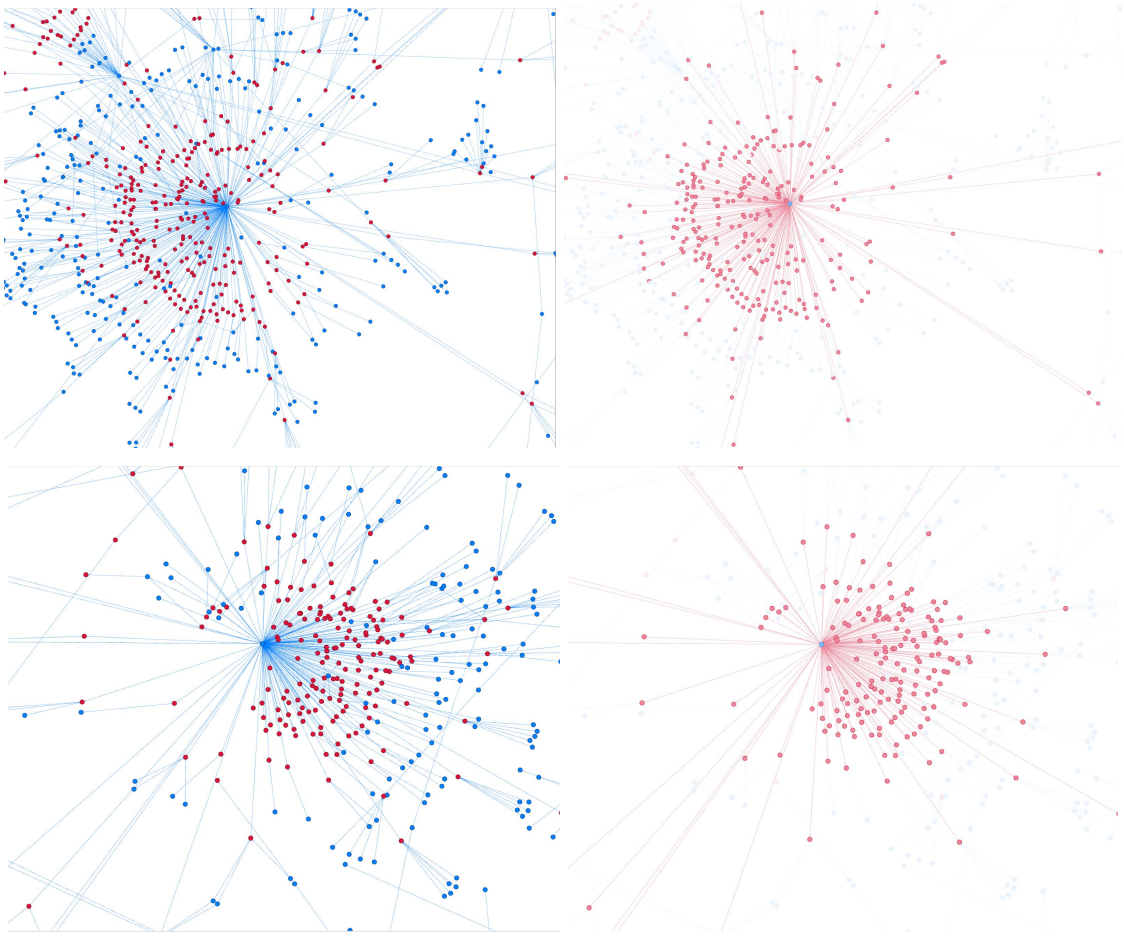

Figure 5.
Dependabot and Renovate

Conclusion

The exploratory analysis indicates that our dataset has a low signal strength (sparse graph) and contains interesting outliers (bots). We should be careful with data cleaning if we are to use this dataset for predictive modeling.