Identification of Distinct Sub-Communities, High-Value Members and Significant Software Packages within the ROS (Robot Operating System) Community

Tony Pratkanis ATP42@STANFORD.EDU
Department of Computer Science
Stanford University

1 Introduction

We conducted an analysis of the network dynamics of the ROS (Robot Operating System) open-source software development community. ROS is a middleware developed for robots by the Open Source Robotics Foundation (OSRF) and Willow Garage, as well as many other institutions and individuals throughout the world. ROS as software can be thought of as a network of packages which can be installed on a computer. Each of these packages contains various nodes, each of which is comprised of components relevant for robots. Examples of nodes include drivers for cameras and other sensors, motion planners, computer-vision detectors, and controllers. These nodes are linked together at run-time through inter-process communication to form a network on the robot's computer system. While we could apply network analysis to this graph, it is quite small and is well understood by ROS software developers. However, the human social community that develops, supports, maintains, and uses ROS and its packages is relatively unexplored. A network analysis of this community would allow us to better understand who is developing ROS and potentially offer insights into how to improve the process.

To do so, we collected a variety of publicly available data from the ROS community as well as obtained data directly from OSRF (which maintains ROS). The data we collected were from three sources. First, we collected the network of packages dependencies – which nodes require the installation of other nodes to function. Second, we collected the source control information for each of the nodes, which included the names of authors and the amount of code contributed by each. Third, we collected information from the ROS Answers question-and-answer website, which included which software the questions referenced, as well as who asked and answered the questions. To perform the analysis, we used the Newman-Girvan (2004) and Clauset-Newman-Moore (2004) methods to identify communities within our networks and find the linkages between them. Following the method of Vasilescu, Filkov, and Serebenik (2013), we also used the technique of comparing source control data and question-and-answer data to produce a more complete set of data for each user. We also used the PageRank algorithm to determine the most significant nodes within the communities we found.

2 Prior Work

Although there have been a number of network analyses of open-source communities, there has been no direct work on the ROS community. However, these analyses of open source networks, especially research on Stack Overflow and GitHub, can provide guidance on how to analyze the ROS community.

First, the work of Anderson, Huttenlocher, Klienberg, and Leskovec (2012) discusses the identification of high-value contributors in Stack Overflow, a programming question-and-answer website. This website allows users to ask technical programming questions (such as “How do I get the current system time in C++?”) and have other users answer them. The Stack Overflow data is similar in format to ROS Answers because the ROS Answers website was modeled on Stack Overflow. Stack Overflow allows users to vote up and down both questions and answers. These votes impact users' reputation scores, allowing the ranking of users on the website. The goal of Anderson et al. is to use network analysis and other techniques to

provide a more detailed view of content value than the simple reputation system provides. The authors found a variety of trends in their data, such as (a) users that answer questions rank highly in the system, (b) questions tend to be answered most by high-ranking users, and (c) answers do not compete for votes in a zero-sum manner. While quite successful in their goals, the work did not consider different Stack Overflow sub-communities, and did not integrate other data sources, such as GitHub into the community. Integrating these data sources could have improved the ranking system by allowing the ranking system to know when an answer was written by the author of the software. This is particularly useful because better answers may be written by the software author who is likely more knowledgeable.

A second study, by Vasilescu et al. (2013), analyzed the relationship and overlap between Stack Overflow users and GitHub users. The researchers wanted to identify if Stack Overflow participation was associated with high developer performance. Would usage of Stack Overflow help the developers be more effective? Alternatively, Stack Overflow could slow down progress by distracting the developers. The authors merged the two datasets and performed statistical analyses of the data to discover a number of facts. First, the authors found that approximately 11% of users in GitHub were active on both GitHub and Stack Overflow. Second, they found that heavy GitHub committers were also heavy Stack Overflow question answerers. Third, heavy GitHub committers also asked fewer questions on Stack Overflow. Fourth, they found that the reverse is partially true – heavy Stack Overflow answerers often commit to GitHub projects. These results, while not entirely network analysis based, are interesting and may be relevant to our study because they serve as a tool for contrast with our results. Specifically, we will be able to determine if these same conclusions hold within the ROS Community.

3 Data and Preprocessing

We collected our data from three sources: the ROS Package graph, the ROS source control system, and the ROS Answers website data provided by OSRF. The ROS Package graph is easy to obtain in machine-readable format from ROS because it is used by the ROS installation system to manage the installation itself. These data formed a directed graph of what packages depended on other packages, becoming what we will call the "package-package graph." The ROS source control system logs were accessed by querying the Git repositories for log data. The total number of commits from each user was added to establish the amount each user contributed to the project. These data were used to construct the "person-package graph." These data were not perfectly clean – some packages (approximately 10%) did not have an associated repository or were not hosted in a Git repository. A manual audit showed that these broken packages were outdated, contained no files, or were unrelated to ROS. Finally, OSRF was kind enough to provide us with the background data from the ROS Answers system. These data included users, questions, answers, and vote counts on posts, as well as some information about which package the questions referenced. This was used to construct three graphs: the "person-question graph," the "person-answer graph," and the "package-question graph." These graphs were merged together prior to analysis.

One challenge for this project was username cross-correlation. The problem was that many users did not use the exact same usernames in Git as they used for ROS Answers, so it was difficult to track down the appropriate user. Further, we were only provided with the numeric user ids in ROS Answers because OSRF wanted to ensure that we would not accidentally receive the database that containded users' passwords. To solve this problem, we found that we could search ROS Answers for the Git email addresses and usernames and develop an approximate understanding of which Git users referred to which ROS Answers users. We then manually audited this generated data to ensure correctness and to add known users. Factors considered in the manual process included first-hand knowledge of account usage, results from Internet searches, obvious typos and in one case directly emailing the account owner. This process, while laborious, ensured the cleanliness of the data we gathered.

4 Graph Model

After collecting our data, we set out to build the graph model. We needed to develop a system where we could adjust and understand the weights of various sub-components of the graphs. Thus, we chose to represent the ROS Community as a weighted, undirected graph. As mentioned previously, there are multiple subgraphs at play in this system. We chose to linearly combine the five subgraphs to produce our final graphs. We determined the weights in our final graph by multiplying each of the weights of the subgraphs by a normalization factor to compensate for the varying scales of the different graphs. The normalization factors were determined by taking the reciprocal of the average weights of each of the subgraphs, so as to normalize the average rates around a value of unity.

In sum, there are three types of nodes in the graph: persons, questions, and packages. There are five types of edges (one type for each of the subgraphs): package-package dependence relations, person-question asked relations, person-question answered relations, person-package developed relations and package-question asked relations. The package-package dependence relations and package-question asked relations are unweighted. The person-question and person-answer relations are weighted based on the sums of votes (upvotes minus downvotes). If a person answered the same question multiple times, then the vote sum was taken over these multiple answers. The package-question graph was simply created by looking at the packages referenced by the questions in the ROS Answers "tagging" system. Some tags did not match the package names directly, so they had to be manually configured, and some tags did not refer to packages at all and were ignored. The person-package graph was populated from the source control repositories, and is weighed by number of commits to the project source code. These five edge types form the total of our graph.

5 Algorithm and Evaluation

Our initial algorithm is based on the one presented by Newman and Girvan (2004) and later improved by Newman (2004). The goal of this algorithm was to identify central clusters within unlabeled networks, such as our ROS graph system. The algorithm is a divisive algorithm, in that it breaks the network down into components by removing edges. The algorithm developed by the Newman and Girvan (2004) operates by removing edges in order of the highest "betweenness." This metric is defined so that it will be largest for edges that bridge communities instead of ones internal to the communities. Since our graph is quite large, we followed Newman and Girvan's recommendation to use shortest path betweenness, which is defined by the number of shortest paths between nodes that pass through a given edge. This can be easily calculated by finding all shortest paths and then determining how many go through a given node. Once these betweenness values are found, the edge with the highest betweenness is removed. This produces a hierarchical clustering of the graph split on edge removal. We would then choose a layer of this hierarchy for the final result which consists of labeling the graph into clusters. While this algorithm is helpful, it is not fully applicable to our work because it does not handle weights.

Fortunately, Newman (2004) has proposed an extension to the algorithm that includes weights. The approach duplicates graph edges based on weighting in the network. This allows the system to handle weighted graphs without issue by simply suggesting that removal of a weighted edge will reduce its weight count by one. Unfortunately, this approach does not handle fractional edges like the ones in our graph. To solve this problem, we will multiply our edge weights by a constant scale factor of 1000 and truncate the decimal place, allowing us to operate the algorithm on our dataset. The Newman-Girvan algorithm works by defining a metric of betweenness with a given edge, defined as the number of shortest-paths between all nodes that pass through it. The edges are then removed from the graph in order of maximum betweenness. Unfortunately, the aforementioned algorithm was too slow to be successful in our project in practice, because our graph was too large. Our graph consisted of over 27000 nodes and thus required over 24 hours to run the algorithm on a high-performance computer (after 24 hours we terminated execution). Since Newman-Girvan never converged, we had to replace it with a faster algorithm.

Luckily, Clauset, Newman, and Moore (2004) describe a different and more efficient algorithm to obtain community structure for networks. This algorithm operates by using a metric known as the modularity. This modularity score Q is defined in terms of two subvalues, defined over each community, and then in terms of a summation of those values:

$$\epsilon_{ij} = \frac{1}{2\mathrm{m}} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, j)$$

$$a_i = \frac{1}{2\mathrm{m}} \sum_{v} \left(\sum_{w} A_{vw} \right) \delta(c_v, i)$$

$$Q = \sum_{i} (\epsilon_{ii} - \alpha_j)^2$$

In these equations, A is the adjacency matrix, delta is a function that is one if its arguments are equal else it is zero, and c is the current community assignment of the ith node. The first value is intuitively the fraction of all edges in the graph that are between the two communities i and j, and the second value is the fraction of all edges in the graph that end in community i. The algorithm proposed is a simple one, in which each node in the network is initially assigned to its own community. Then, a greedy algorithm is run in which all possible node community transfers are considered and the one that increases Q the most is selected. This process is repeated until we cannot increase Q anymore, at which point we have completed the process. It may seem that calculating the changes in Q would be slow due to the large number of possible actions at each step, but it is possible to use a mathematical simplification to reduce the overhead of calculating the change in Q to a reasonable amount. We used the igraph library to perform this process (Csardi and Nepusz, 2006). This algorithm ran quickly on our data (in under 5 minutes), and as discussed below, produced satisfactory results despite its approximate nature.

Once we developed the list of clusters in our data, we applied the weighted PageRank (Xing and Ghorbani, 2004) algorithm to score the results within the communities, by applying it to the subgraph of the nodes and edges within the community. We used the PageRank output to identify the most important nodes, and reported the top most nodes of each of the three types (package, user and question). We used basic weighted PageRank, although because our graph was undirected, we treated each edge as two directional edges, one in each direction for the purposes of the algorithm. The weighted PageRank algorithm works by defining a simple function:

$$R(u) = (1 - d) + d \sum_{v} R(v) W_{uv}$$

In this equation. R is a function of each node, d is the empirical decay factor (usually 0.85), and W is the weight of the link from node u to node v. Since R is defined recursively, all R values are set to a constant initial value (1 / n, were n is the number of nodes) and then the algorithm is executed iteratively until convergence. This usually takes about 100 iterations. In addition, after each iteration, the values of R have to be normalized. That is, they must be divided by their sum so that they are made to sum to one. After the algorithm has converged, the values of R represent the ranks of the individual nodes. The nodes with the largest R value represent the most important nodes in the network, in that they have the most other nodes linked to them.

To evaluate the results, many self-consistency checks can be used for self-validation. First, we engage in the most obvious form of evaluation, which is to compare the generated list of most valuable ROS users with the list of high-ranking ROS Answers users. We will validate the results by inspection, understanding what has been ranked highly and why, and justify these results. We also looked for any particularly glaring exceptions in our data, given our first-hand knowledge of the ROS community.

6 Results

Table 1 lists some basic statistics about the ROS community database. It includes all data from Git until November 10, 2014 as well as all data from ROS Answers until October 28, 2014.

Table 1: ROS Community Summary Statistics

Value	Result
Total Number of Nodes	27559
Total Number of Edges	59763
Number of Packages	2142
Packages with Valid Repositories	1919
Number of ROS Answers questions	19178
Number of users*	6462
Number of users only on Git	321
Number of users only on ROS Answers	6141
Number of users on both ROS Answers and Git	269
Average number of commits per user per package**	7.041264
Average number of commits per user	8.23878
Average number of packages changed per user	1.17007
Average number of questions asked per user	2.9688
Average number of questions answered per user	0.11966
Average number of questions asked per package	7.16519
Network Average Clustering	0.001517
Network Diameter (of largest connected component)	16
Percent of nodes in largest connected component	91.17%

*A user is defined as someone who has taken at least one action – to have asked or answered a question or to have made at least one commit. This ignores the many ROS Answers accounts that exist only for the purposes of spam or are for lurkers.
**Includes only users who have committed to the project in question at least once.

After collecting this initial data, we ran the community detection algorithm and then ran PageRank on the resultant communities to identify the top-ranked nodes in each of the communities. There were a total of 966 communities extracted by the system, of which 622 were considered to be non-spam (spam "communities" are simply a single user and one or many questions unrelated to ROS, often reminiscent of email spam). Of these 622, an additional 467 consisted of only a single user and/or a question or package. These 467 "communities" represent users who maintain packages based on ROS that are not widely used (such as student projects) or users who post a single question on ROS Answers and never return to the site. Further, 99 of the remaining 155 communities consist of less than 10 nodes, mostly users and questions. Each of the remaining 56 largest communities has many users, questions, and packages. There was a general trend that the largest of the communities consist mostly of questions and users, while the smaller communities contained mostly users and packages.

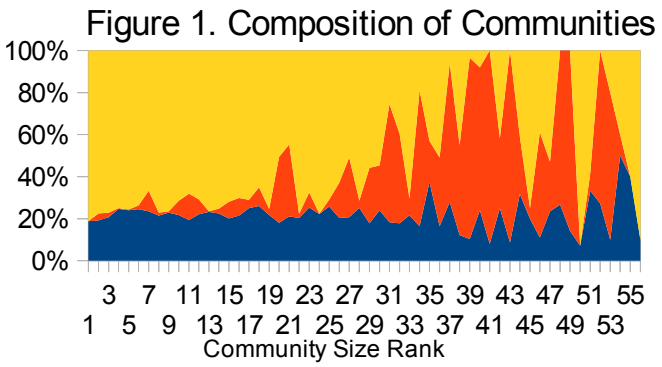
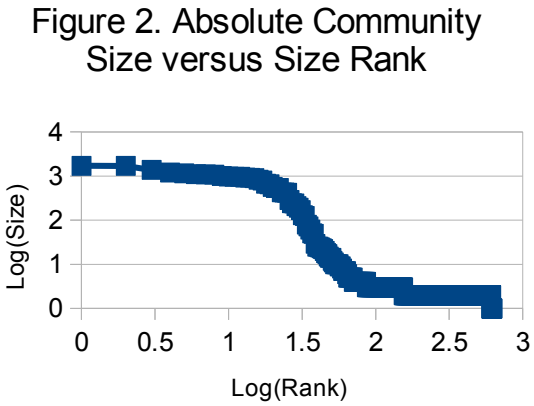


Figure 1 shows how community content changes with size rank for the 56 largest communities. The blue is users, the yellow is questions, and orange is packages. The community size rank is the order of community size (i.e., a rank of 1 is the largest, 2 is the 2nd largest, etc.). The graph shows how smaller communities generally consist of more packages than questions.



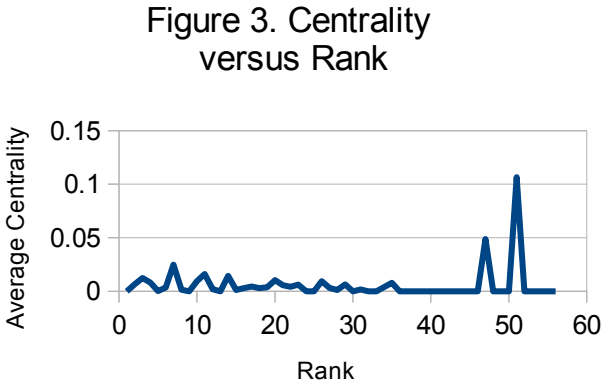
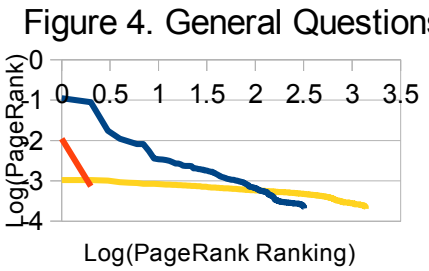
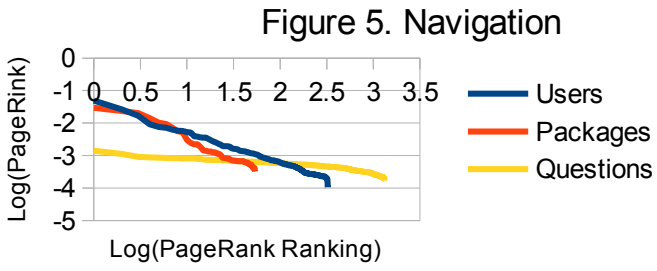


Figure 2 shows a log-log plot of the size of the communities versus size rank for all communities. The graph shows that there are a few similarly-sized large communities and many smaller communities.

Figure 3 shows the average centrality of the communities versus rank. Some of the communities have low or zero centrality scores. These communities were found to be a distributed set of people asking and answering questions on ROS answers. The communities with large centrality scores were found to be packages with a central developer or group of developers and other users around them.





Figures 4 and 5 show the ordered PageRank scores of the nodes of each type in the two highest-ranking communities. PageRank scores represent how valuable a node is. In the case of users, this is a function of how many commits the user has made and how many questions the user asked and answered. For packages, it is a function of how many questions were asked and how many commits were made. For questions, it is a sum of the votes on the question and the answers. The PageRank scores show that, in the more decentralized community ("Generic Questions"), the user PageRank graph is initially more level than in the centralized "Navigation" community.

Figure 6. ROS Inter-Community Structure

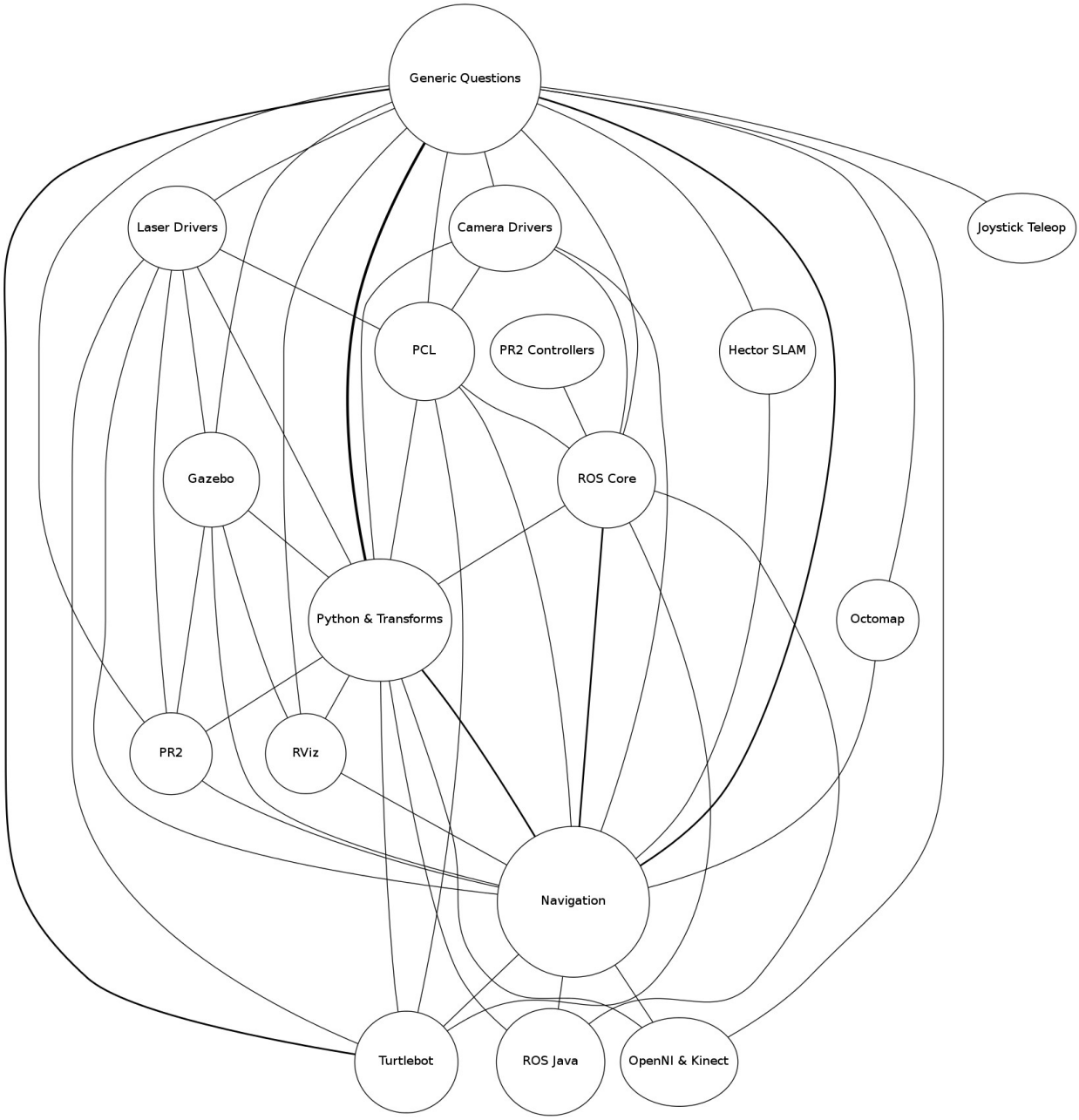


Figure 6 shows the linkages between communities with over 700 members (17 communities), to which names have been manually assigned. The width of the lines in the graph represents the sum of all the weights of all edges between nodes in the two communities. Because the graph is essentially complete, we only show the top 50 lines in the graph. One fact that can be discovered is that the largest line from most communities is the line to the "Generic Questions" community. The communities that violate this rule are "ROS Java" and "PR2 Controllers" which are populated with more advanced ROS users who are less likely to be interested in generic ROS questions.

Table 2 lists the largest ten communities in terms of number of nodes. It also lists the top-ranked packages and categories in the PageRank output. The ranks of the users were also gathered, but are not reported directly for privacy reasons. The "community name" column is a human-readable name assigned by the author of this paper manually to the detected community.

Table 2: Largest Sub-Communities within ROS

Com -munity	Clus-tering	Total	Users	Pack -ages	Ques-tions	Top Packages	Top Questions
Generic Questions	0.000	1702	319	2	1381	rosmake, pr2_2dnav_slam	"How to extract data from *.bag?"
Navigation	0.006	1694	326	54	1314	navigation, move_base, gmapping, amcl	"Voice commands / speech to and from robot?",
Python & Transforms	0.012	1377	286	29	1062	tf, rospy, nodelet, geometry, geometry_msgs	"When should one use answers.ros.org vs. the ros-users mailing list?"
ROS Java & Android	0.008	1204	297	5	902	rosjava, android_core, rosjava_core	"Building rosjava fails at task ':apache_xmlrpc_common:compileJava'"
Turtlebot	0.000	1147	275	4	868	turtlebot, ros, calibration	"Can't access turtlebot through ttyUSB0"
PCL	0.003	1105	271	19	815	pcl, stage, pcl_ros	"ROS installation on Mac with Homebrew--bootstrapping issues?"
ROS Core System	0.025	1086	255	106	725	catkin, rqt, rosbuild, roscpp	"Catkin and eclipse"
Gazebo	0.001	1073	231	14	828	urdf, gazebo_plugins, pr2_simulator	"Significance of ros::spinOnce()", "new to open source"
OpenNI & Kinect	0.000	1000	229	6	765	openni_launch, openni_tracker, pi_tracker	"openni_launch not working in fuerte + ubuntu precise 12.04"
Camera Drivers	0.009	973	211	67	695	camera1394, velodyne, velodyne_driver	"rosdep: command not found"

The largest community consists mostly of questions and users and has been named "Generic Questions." These questions are of a type that apply to all ROS nodes, such as "How to Extract data from *.bag?" - a question asking how to use one of the data log files stored by the ROS system. This group also contains many different users, thus leading to its zero average clustering coefficient. The "Navigation," "Python & Transforms," and "ROS Java" communities are examples of work centered on a specific task. The "Navigation" community contains the packages "navigation," "gmapping," and "AMCL" which are all part of the navigation components of the ROS system. These communities have a non-zero clustering coefficient because they are focused around a few individuals who developed the software. The "Python & Transforms" community contains many tools used for mathematics and linear algebra in ROS. The "ROS Java" community is also of a similar type, containing only packages associated with the Java bindings for ROS and applications developed with those bindings (such as ROS Android). While we chose not report the user data for privacy reasons, we are able to confirm that the highly ranked users involved in these

communities are the often full-time developers of these tools and packages, and are often highly ranked on the ROS Answers system website.

7 Validation

We engaged in several steps to confirm our results. First, we identified top users by PageRank in the largest communities. As expected, these users were found to be users that contributed a great deal to ROS Answers, and therefore were all found on the ROS Answers website as top contributors. In communities clearly centered around a package (such as the "Python & Transforms" community), we found that top-ranked users were likely to be professional developers of the packages in question. Additionally, the highly ranked questions were found to be popular on the ROS Answers site.

8 Discussion & Future Work

Our results confirm the hypothesis that the ROS community consists of many different sub-communities. Further, our results demonstrate that these sub-communities are often focused around a specific set of subcomponents or applications of the ROS system, or around the ROS answers system itself. This can be shown by examining the top communities ranked by size.

We found that some communities were based around questions and answers (such as the "Generic Questions" community) and some were based around packages (such as the "Navigation" community). The question-only communities are more distributed, and thus have lower (or zero) clustering coefficients, while the package-based communities have a larger value. We also found a trend in that smaller communities consisted of fewer questions as a percentage of nodes by type (see Figure 1). This effect is likely explained by the result that many smaller communities consist of developers working in a single lab or company (for example one group consisted mainly of employees of Rethink Robotics and was devoted to the companies' products) and therefore these developers do not use ROS Answers as a communication tool, preferring other options such as face-to-face communication or internal mailing lists to discuss technical issues.

While our project was largely successful, there are three problems that become apparent. The first is that some heavy developers often work on several independent projects. For example, the author of this paper worked on both the Gazebo simulator and the ROS Java project. This can lead to the fusion of two communities that would appear distinct to an outside observer (such as ROS Python and the TF Library) but that are developed by largely the same developers. The second problem is that there are many generic questions that are applicable for all ROS users (for example, "How do I install ROS on Mac?") that are often answered by leading users in communities. Since these questions get a lot of upvotes, they become the top ranked questions in those communities. Third, there are some packages that are not associated with any questions, because ROS Answers users did not ever ask questions about those packages or did not tag them in the questions they asked. This is common if there are a number of packages related to a specific robot (such as "pr2_2dnav_slam" instead of just "2dnav_slam") for which the generic package is tagged. Thus, these wayward packages can be assigned to the wrong community because they are only linked to their developers and not to their users. While these three problems are evident in the data, the general ranking and membership information is still useful for human analysis of the ROS ecosystem.

We believe that our results will be useful to the ROS community and OSRF as they will allow us to understand the various developers and social groups at work developing and using ROS. For example, if OSRF wanted to create a group of developers who could help direct the future of ROS development, they would do well to choose high-ranking developers from each of the different ROS development groups identified in our study. This could also help prevent the feared fragmentation of the ROS community along the different development groups. The network analysis conducted in this paper could be performed at various intervals to detect changes in the ROS communities that could then be reflected in the membership of the directing developers.

Our results are also useful to the general network analysis community because they demonstrate how network analysis techniques can be applied to extract valuable information. Specifically, they demonstrate the applicability of common analysis techniques such as centrality, community detection, and PageRank can be successfully applied to networks with many different types of nodes (in this case users, questions, and packages). Further, our work can be contrasted with previous work on question-and-answer sites and their relationship to software development. Specifically, we found similar trends in that a few high-ranking users answer the majority of the questions (see Figures 4 and 5) and do the majority of development work. These users are active in both question-and-answer sites and software development, as in Vasilescu et al. This replication suggests to the analysis community that these trends might apply to all similar open-source development systems, not just to StackOverflow.

In terms of future work, we first plan to produce a version of our system that can be used repeatedly to identify what communities and users are relevant over time. We might also work to resolve the three deficiencies above. The problem of community merging could be solved by developing an approach where a node could be in multiple communities so that a developer on working on both projects could be a member of both communities. Another approach to this problem might be to develop a method of splitting contributions over time, so that a developer working on one project and then on another would be distributed over the two. Finally, we could develop a system using natural language processing to improve the tagging system, so that packages and questions could be better related and generic questions could be categorized as applying to the entire system.

References

A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 850-858, Beijing, China, 2012.

A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. In *Physical Review E*, 70, 066111, 2004.

G. Csardi and T. Nepusz. The igraph software package for complex network research. In *InterJournal, Complex Systems*. Page 1695. 2006. http://igraph.org/

M. E. J. Newman. Analysis of weighted networks. *Physical Review E*, 70(056131), 2004.

M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113), 2004.

B. Vasilescu, V. Filkov, and A. Serebrenik. StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge. In *Proceedings of the 2013 International Conference on Social Computing (SocialCom)*, pages 188-195, Alexandria, Virginia, 2013.

W. Xing and A. Ghorbani. Weighted PageRank algorithm. In *Proceedings of the Second Annual Conference on Communication Networks and Services Research*. Pages 19-21 2004.

J. Yang, J. McAuley, and J. Leskovec. Community Detection in Networks with Node Attributes. In *Proceedings of the 2013 IEEE 13th International Conference on Data Mining (ICDM)*, pages 1151-1156, Dallas Texas, 2013.

