Network Semantic Segmentation with Application to GitHub

Neda Hajiakhoond Bidoki
Department of Computer Science
University of Central Florida
Orlando, FL
Email:nedahaji@cs.ucf.edu

Gita Sukthankar
Department of Computer Science
University of Central Florida
Orlando, FL
Email: gitars@eecs.ucf.edu

***Abstract*—In this paper we introduce the concept of *network semantic segmentation* for social network analysis. We consider the GitHub social coding network which has been a center of attention for both researchers and software developers. Network semantic segmentation describes the process of associating each user with a class label such as a topic of interest. We augment node attributes with network significant connections and then employ machine learning approaches to cluster the users. We compare the results with a network segmentation performed using community detection algorithms and one executed by clustering with node attributes. Results are compared in terms of community diversity within the semantic segments along with topic coverage.**

***Index Terms*—Social Networks, GitHub Social Coding, Community Detection**

I. Introduction

Social media platforms serve as powerful tools that enable people to share information and collaborate on tasks such as the development of software code [1], [2]. They can also serve as a valuable laboratory for understanding social behavior and teamwork in human groups. One of the most popular social coding platforms is GitHub. It hosts many development projects and provides a space for professionals who work together. GitHub is unique compared to other social networks since developers with different backgrounds, interests and levels of proficiency interact with each other to engage in asynchronous team-based software development. This provides an opportunity for researchers to investigate team-based community evolution. Recently, there have been many studies on GitHub. Authors in [3], [4] investigate key drivers and behaviors in GitHub projects and teams. Other studies [5]–[8] investigate the impact of utilizing this platform on the software development process. Much of the work in this area employs data-driven approaches to investigate social behaviors or teamwork communication [9]–[11]. In addition to data-driven techniques, there are also several examples of survey and interview studies [12]–[14].

In this work we evaluate the performance of network semantic segmentation on the GitHub social coding network to segment the network into the sections according to repository topics, such as machine learning, algorithms, game development, etc. We employ users' attributes alongside with the network connections to group the GitHub users. This type of network segmentation can shed light on important social network analysis questions such as the evolution of topic areas. This paper compares three segmentation approaches. First, community ensemble combines multiple communities into a single segment. Second, to execute user-network attributes clustering, we augment node attributes with significant connections for network semantic segmentation. We then employ machine learning approaches for clustering the users [15]. The final clustering approach relies only on users' profile attributes to classify them. Our evaluation was conducted on a dataset composed of the public GitHub events and repository profiles from 2015 through 2017.

The remainder of this paper is organized as follows. Section II describes our dataset. Section III and IV state the problem and our method respectively. Results are provided in Section V and then we conclude the paper in Section VI.

II. Dataset

For this study, we extract communities from the GitHub online social network. With almost 20 million users and 57 repositories, GitHub is the largest host of source code in the world. Before describing the data in detail, we will introduce some terms.

Definitions:

- **Repository:** is a location where a particular project and all files associated with it are stored. It is usually abbreviated to repo.
- **Event:** refers to the various activity streams on GitHub. Developers have different kind of interactions with repositories such as pulling, pushing, committing, etc. These events are considered as the links between user and repository nodes.
- **User:** refers to any person or organization who has created a profile on the GitHub platform.
- **Topic**: represents intended purpose, subject area, affinity groups, or other important qualities of the repository related to a project.
- **Community** refers to a group of users who interact frequently with the same repositories by creating different types of events.

We mined the community structure from January 2015 until January 2017. This study focuses on the top 1000 repositories. To retrieve these samples, repositories were ordered by their corresponding events. We then selected 1000 unique repositories, along with the users who interact with them. We extracted the events corresponding to the users whose information is publicly available (about 8437). These events were then used to create the repository-user network. The resulting network was a bipartite graph with repositories in the first set and users in the second set. This network is then projected on the user nodes to create the user network. With the new graph we were able to capture the user communities with the Louvain community detection algorithm, an efficient method for identifying communities in large networks.

III. Problem Statement

Identifying network segments that consist of groups of users with the same research area or interest is our goal. Mathematically, we are addressing the following problem.

Problem definition:

We consider $G = (U, V, E, C, T_t)$ to denote a bipartite graph whose partition has the parts U and V, and E denotes the edges of the graph. C is a $|C| <= N$ communities associated with users. Each user belongs to a community. Our goal is to split the network semantically into T segments where each of them consists of users associated with a class label $t \in T$ representing a set of topics. Fig. 1 shows a small scale GitHub network. Users interact with different repositories; each repository is associated with multiple topics.

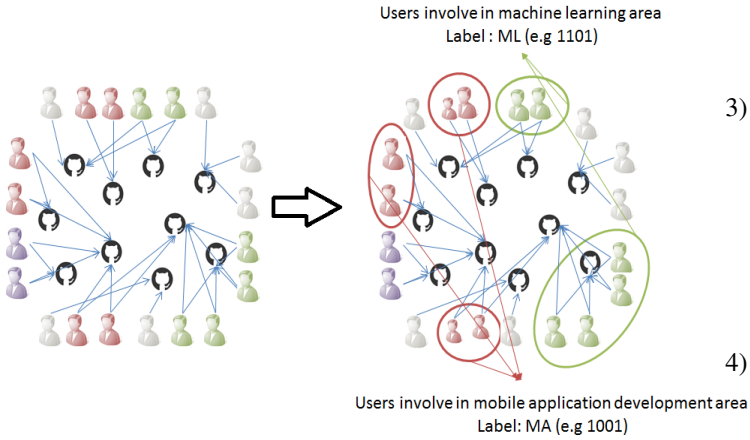

Fig. 1. Github Semantic Segmentation - Users are labeled with labels corresponding to their activities

IV. Method

To address the problem stated in section III we propose and investigate the performance of three approaches:

- **Community ensemble**: This approach combines multiple communities which have been already detected by community detection algorithms into a single segment. This approach is common among consensus solutions. We employ hierarchical agglomerative clustering (HAC) for this purpose. We labeled each community member with the topic common among all the members. The distance of communities with the same labels is zero. Otherwise they are combined according to the distance between their labels.
- **User-Network attribute clustering**: This approach uses the regular clustering algorithms to apply the semantic segmentation in such a way that they are similar in terms of topic attribute as well as network structure, meaning that they interact with similar repositories or belong to the same community. We use the most active repositories as the categorical attribute for users.
- **Classic clustering**: This method is the standard clustering approach using the users' attributes to classify them.

The following procedure was used to find the semantic segments on GitHub:

1) **Building the graph**

   The corresponding graph was created using the NetworkX package. We used anonymized users and repo ids to feed into the NetworkX graph as nodes. Our target graph consists of repositories and users as U and V sets respectively and events in our study dataset that form the edges of the network, represented as E.
2) **Extracting network structure attribute** The well known repositories can be a measure to categorize the users. For example if some users frequently interact with TensorFlow (an open-source data flow programming library commonly used in the machine learning area), they can be be grouped into one category if they are close enough in terms of other attributes as well.
3) **Network Graph Projection** To detect the community of users we employ a graph projection method on the user-repository bipartite graph. Bipartite network projection is an broadly used technique for condensing information about bipartite networks. By this way two users interacting with the same repository connect to each other via an edge. Figure 2 illustrates how we have created the user graph from the original bipartite graph.
4) **Community Detection** After creating the user network graph, we applied the Louvain algorithm to determine the actual communities that exist in the current network. The Louvain method optimizes the modularity of the detected communities. With this approach the quality of a community assignment is measured and quantified by the density of the connections which exist within communities compared to a particular type of random network. Until here we have detected the communities that users are associated with and have labeled each user with its community label.
5) **One Hot Encoding** Those attributes that we use are categorical features. Machine learning algorithms often

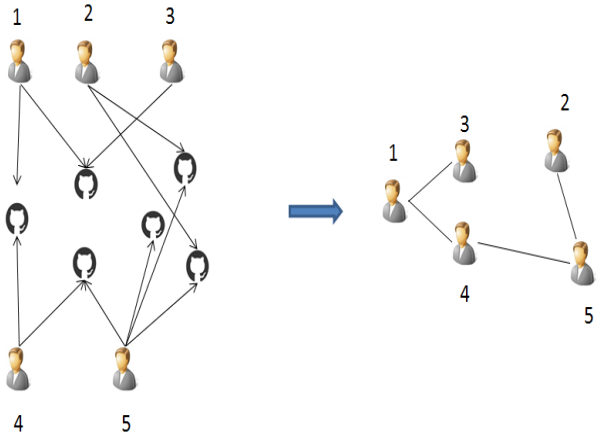

Fig. 2. GitHub Network Projection Example - Bipartite user-repo graph is converted to user only graph

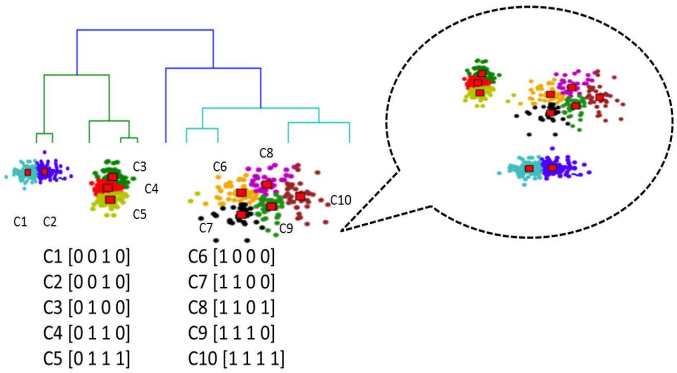

Fig. 3. Agglomerative approach - communities are grouped hierarchically and their members' labels are defined with regard to the whole population of the new community

cannot use categorical data directly. Also, there is no ordinal relationship between the features we have selected. Therefore, we have used one hot coding for the topic features.

6) **Unsupervised Clustering**
After we extracted the network features of target nodes and added them to the node attributes as additional ones, we then employed clustering techniques. Clustering algorithms are used to find similar groups in data. As we are taking advantage of network features alongside with other features, similarity would be measured based on both advanced features as well as the other attributes such as topic of interest. We have tried multiple clustering algorithms such as Mini Batch KMeans, MeanShift, Ward, Birch and Agglomerative Clustering. We finally selected Mini Batch KMeans and Birch for regular clustering approach and Agglomerative Clustering for community ensemble based on the precision and time complexity of these algorithms.

V. Results

To evaluate the methods we have focused on the diversity of communities in the segments after using each approach. All three approaches group the users into the segments which are categorized as different topics. However, not only it is important that users be grouped into the categories with the true topic label, but also it is significant that they belong to the semantic segment with whom they share the greater number of interactions. Diversity is the quantitative measure that measures how many different types of communities there are in each semantic segment. There are multiple metrics to evaluate the diversity of members inside a semantic segment. The Shannon index, also known as Shannon's diversity index or Shannon-Wiener index, is a popular method to quantify the entropy. Shannon–Wiener diversity index is defined as:

$$H = -\sum_{i=1}^{s} p_i \times \ln(p_i) \tag{1}$$

where s is the number of communities and p_i is the proportion of the semantic segment represented by community i. We have used python skbio package in order to calculate the Shannon index.

Another metric that we have considered to compare the approaches is the total number of topics covered by each algorithm. Figures 4-7 show the comparison results.

As Figure 4 shows, the agglomerative approach has a lower diversity of entropy, indicating that users in each segment have a higher degree of interaction. They belong to a narrower range of communities and therefore they mostly highly interact with the other users in the same segment. Birch grouped the people from a wider range of communities into segments. Users of each segment have similarities in terms of topic of interest or activity however they belong to wider range of communities and have less interactions with the users of the same segment. Mini Batch K-means captured the segments with an even greater range of communities compared to the other two approaches. On the other hand, the agglomerative approach covers a smaller range of topics. This is due the fact that we use a majority voting approach to label the communities and segments as the hierarchy goes up. Birch covers a wider range of topics and Mini Batch K-means covered the highest range of topics. Having network structure as attributes alongside with topic attributes could decrease the entropy while capturing more topics. Therefore, it outperforms the agglomerative algorithm in terms of topic coverage and clustering based on topics alone in terms of entropy.

VI. Conclusion

We introduced the concept of network semantic segmentation as a useful tool for social network analysis. This paper compared three approaches for the segmentation. First, community ensemble combines multiple communities into a single segment. Communities are grouped with regard to common topics across all their members. The second approach executes user-network attribute clustering using the regular clustering

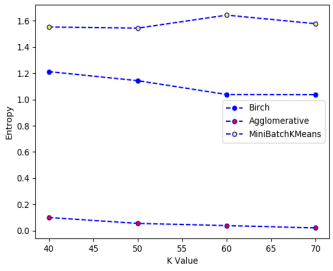

Fig. 4. Semantic segment entropy

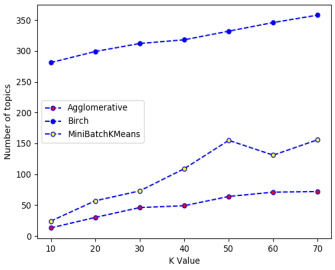

Fig. 5. Semantic segment topics coverage

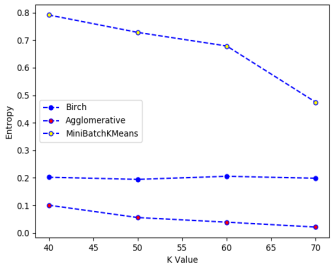

Fig. 6. Semantic segment entropy - having network structure attributes alongside with topics in clustering

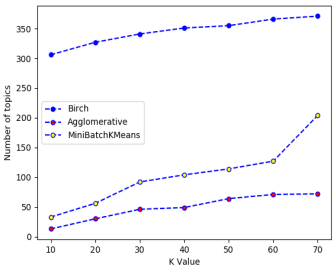

Fig. 7. Semantic segment topics coverage - having network structure attributes alongside with topics in clustering

algorithms to group the users into segments based on their similarity in topic as well as network attributes. It uses the most popular repositories as categorical attributes for users. The third method is the clustering approach which uses the users' attributes to classify them. We employed the above methods on a dataset consisting of three years of GitHub data.

ACKNOWLEDGMENTS

This research was supported by DARPA program HR001117S0018.

REFERENCES

[1] A. Begel, R. DeLine, and T. Zimmermann, "Social media for software engineering," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 33–38.

[2] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng, "The impact of social media on software engineering practices and tools," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 359–364.

[3] B. Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P. Devanbu, and V. Filkov, "Gender and tenure diversity in github teams," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 2015, pp. 3789–3798.

[4] F. Calefato, F. Lanubile, and N. Novielli, "A preliminary analysis on the effects of propensity to trust in distributed software development," in *Global Software Engineering (ICGSE), 2017 IEEE 12th International Conference on*. IEEE, 2017, pp. 56–60.

[5] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Social computing (SocialCom), 2013 international conference on*. IEEE, 2013, pp. 188–195.

[6] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 345–355.

[7] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 155–165.

[8] B. Vasilescu, A. Serebrenik, and V. Filkov, "A data set for social diversity studies of github teams," in *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 514–517.

[9] D. M. Soares, M. L. de Lima Júnior, A. Plastino, and L. Murta, "What factors influence the reviewer assignment to pull requests?" *Information and Software Technology*, vol. 98, pp. 32–43, 2018.

[10] K. Aggarwal, A. Hindle, and E. Stroulia, "Co-evolution of project documentation and popularity within github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 360–363.

[11] C. Casalnuovo, B. Vasilescu, P. Devanbu, and V. Filkov, "Developer onboarding in github: the role of prior social links and language experience," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 817–828.

[12] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on computer supported cooperative work*. ACM, 2012, pp. 1277–1286.

[13] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: activity traces and personal profiles in github," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 117–128.

[14] B. Vasilescu, V. Filkov, and A. Serebrenik, "Perceptions of diversity on github: A user survey," in *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE Press, 2015, pp. 50–56.

[15] N. H. Bidoki and M. B. Baghdadabad, "A glance at structural classifiers and their applications," in *2nd National Conference on New Approaches in Computer Engineering Islamic Azad University, Roudsar-Branch National Conference*, 2016.