Finding Communities in Stack Overflow Questions

Jake Carlson

March 14, 2019

Abstract

A report on implementing algorithms to find community structure in large graphs. We will explore several different methods for finding communities, including modularity and Walktrap. I implement each of these methods in Python and show how they perform on Stack Overflow questions and answers.

Contents

1   Introduction

Finding communities in network data has become more important in recent years. Community detection allows researchers to examine connections between people, and communication between different goups of people. This sort of analysis can provide deeper insight into how certain social networks operate, and can help explain some real-world events. The widespread usage of social networks has made large, graph-based data sets widely available that provide fine-grained information about how people connect and interact.

It is possible to find communities within this graph data by defining what a community is in a graph, and then finding a way to extract that information from the graph data set. Community structure is typically defined as densely connected groups of verticies, joined by sparser connections between groups [2]. This definition is more formally encoded in the idea of graph modularity. The modularity of a graph increases if there are fewer than expected edges between communities. The expected number of edges between communities is based off of how many edges one would expect to fall between communities if edges were inserted at random [3]. This concept is discussed more in the Modularity section.

2   Background

The usefulness of community detection was demonstrated following the 2016 U.S. Presidential Election. Since so many models of candidate support proved to be incorrect, there was a scramble to find how so many supporters were not included in the building of those models. One social network that was studied was Twitter. Researchers from the Electome project at the MIT Media Lab performed an analysis of tweets leading up to the 2016 election. They found groups surrounding key issues, and individuals who were central to the conversation about those issues. By analyzing the follower networks for those users, and examining how those follower networks overlap, the researchers were able to find community structure in the network that proved critical to the outcome of the 2016 election [6].

The network structure can clearly be seen in 1. In the figure, several clear groups are dsitinguishable, several in the upper left of the graphic, and one in the bottom right. Here, the groups in the top left are groups that supported Hillary Clinton, and the group in the bottom right supported Donald Trump. It can be seen that even between the different groups surrounding Clinton, there are still edges connecting users between those groups. This is in contrast to the major group surrounding Trump. Trump supporters had very few connections to people outside of the group, and key influencers and reporters had very few connections to the Trump group. This could explain why so many journalists were surprised by the result of the 2016 election, because they had no grasp over the size of the Trump support network.

It is important to note that the MIT Media Lab researchers based their network analysis off of finding individuals central to the discussion of key issues. They used the PageRank algorithm to find these key individuals, and then looked at the overlap of their follower networks to find the community structures. Examining subgraph overlap is a key part of other community detection algorithms, including Walktrap, which will be discussed in depth later.

3   Algorithms

Several algorithms have been introduced to mine communities from connection information. Several of these algorithms are discussed here, along with details of my reimplementation of these algorithms.

3.1   Spectral Bisection

The spectral bisection method for detecting communities in graphs proposed by Pothen et al. involves using the eigenvectors of the Laplacian matrix for a graph to determine where to cut the graph [5]. The Laplacian matrix for a graph is given by:

$$L = D - A \tag{1}$$

where D is the diagonal matrix containing the degrees of each vertex, and A is the adjacency matrix for the graph. Once this matrix is found, the graph can be divided by taking the median value x of the eigenvector with the second lowest eigenvalue of the Laplacian matrix. The graph is then split into
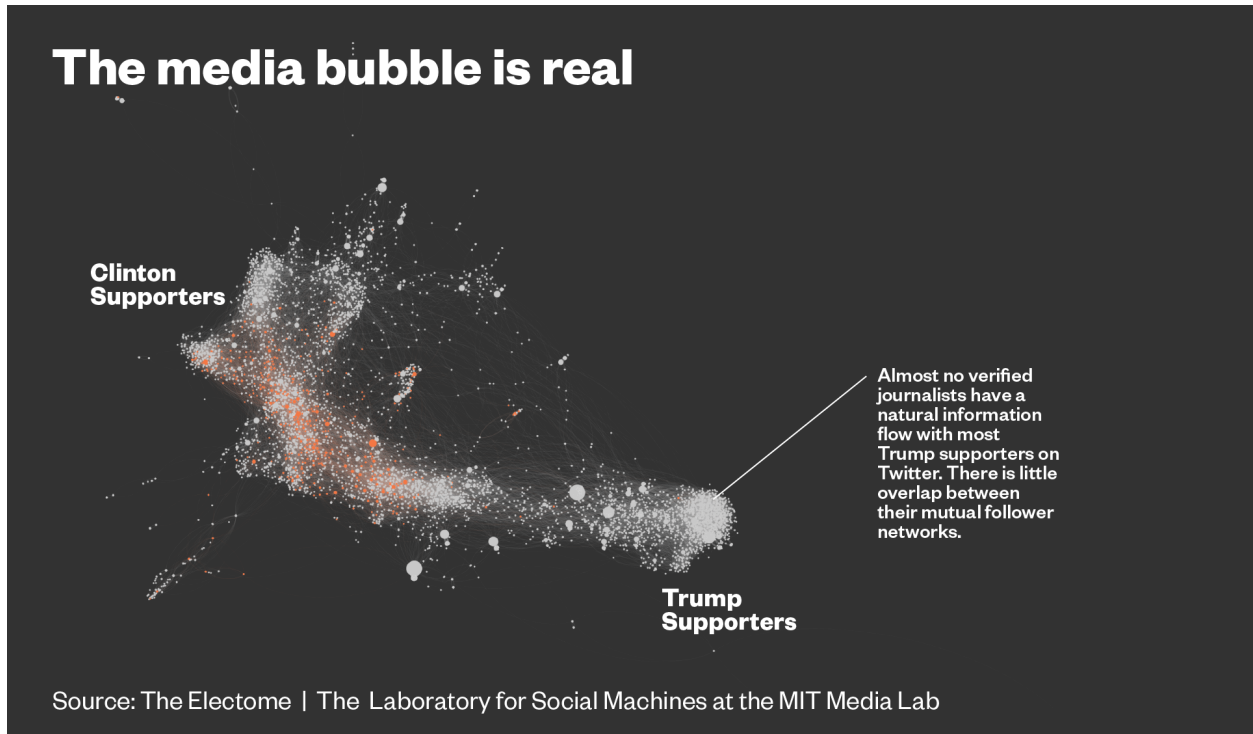
Figure 1: Twitter follow networks for key influencers in the 2016 U.S. Presidential Election

two sets, A and B where A has verticies whose components are less than or equal to x, and B has all vertices whose components are greater than x. If the difference in the sizes of these sets is greater than one, verticies whose components equal x are arbitrarily reassigned to B until the difference in the sizes of A and B is one.

We then consider the set of edges who have one vertex in A and another vertex in B. This is an edge separator of the graph, taken from the bipartite graph contained in A and B. With this edge separator, it is possible to compute the minimum vertex cover of the bipartite graph contained in A and B. This cover is then used as the vertex separator to divide the graph into two parts.

This method works well when the graph is easily separable into two distinct groups, however, it struggles when the graph is not easily separable [2].

3.2   Edge Betweenness

In 2002, Girvan and Newman developed a new partitioning method by looking at the edges that are least central to communities. This development was motivated by shortcomings of a heirarchical clustering algorithm when applied to community detection. The issue Girvan and Newman found was that the heirarchical method would often take a vertex that shared a single edge with a community, and split this vertex into a community of its own. Girvan and Newman thought it was intuitive that a vertex that has only one edge should belong to the community that contains the vertex at the other end of that edge [1].

To account for this, Girvan and Newman developed a new method for partitioning a graph based on looking for edges that are the most "between" communities, rather than looking at those that are most central to the communities as heirarchical clustering does. Instead of iteratively adding the strongest edges to a set of verticies, the new method iteratively removes edges from the original graph. Girvan and Newman extend Freeman's betweenness centrality to determine the edges that are the most between communities.

Freeman's betweenness centrality is based off of finding the shortest path connecting two nodes. More formally, the betweenness centrality for some vertex k is the number of shortest paths between all other pairs of verticies i and j such that the shortest path between i and j contains k. Girvan and Newman generalize this to apply to edges, where the betweenness of an edge is defined as the number of shortest paths that run along that edge.

Once the betweenness has been calculated for all edges in the graph, the edge with the highest betweenness is removed. The betweenness is then recalculated for the new graph, and the next edge with the highest betweenness is removed. This process repeats until there are no more edges in the graph.

3.3 Modularity

Using modularity to detect communities in graphs was first proposed by Newman in 2006 [3]. Newman identified that many community mining methods assume the network divides appropriately into subgroups, however, real-world social networks rarely behave this way. Instead, users can belong to several communities. In addition, typical graph partitioning algorithms are not suited well for community detection. These methods usually use minimum cut to divide the graph into subgroups. This is an intuitive method to apply, as it will minimize the number of edges running between the communities. However, you may need to constrain the size of the communities to prevent trivial solutions to dividing the network. For example, if the goal is to minimize edges between groups, the most straight-forward solution would be to create two groups, one of which is empty and one of which contains the rest of the network. This produces a count of edges between groups of zero. This trivial solution can be avoided by saying there needs to be at least k nodes in each subgroup, but this requires determining this value ahead of time. Because of this, Newman was motivated to develop a new community detection method that is more applicable to real data.

One goal Newman had was to eliminate the dependency on constraining the subgroup sizes as is required by typical graph partitioning algorithms. This would avoid the problem of trivially dividing a network into two subgroups where one is empty and the other contains the rest of the graph. To achieve this, Newman introduces the modularity matrix. The modularity matrix is defined using the following equation for each element of the matrix [3]:

$$B_{ij} = A_{ij} - E_{ij} \tag{2}$$

$$E_{ij} = \frac{k_i k_j}{2m} \tag{3}$$

Where A_{ij} is the count of the edges between nodes i and j, and E_{ij} is the expected number of edges between the two nodes, given that k_n is the degree of some node n and m is the total number of edges in the network. One important property of this matrix is that all of the rows and columns sum to zero, meaning it has an eigenvector $(1, 1, 1,...)$ with an eigenvalue of zero. This is similar to the Laplacian Matrix from the Spectral Bisection Method. This matrix is then represented as a linear combination of normalized eigenvectors.

The method for dividing the network into groups is borrowed from Spectral Partitioning. Newman repeatedly selects the largest, positive eigenvalue as the basis for dividing the network. The network is then divided based on the values in the eigenvector, where positive values are grouped together and negative values are grouped together. It can also be noted that the magnitude of these values indicate the strength of which an element belongs to a subgroup. Large values indicate an element belongs with a group, while values close to zero indicate the element may fall between the two groups.

This process continues until there are no more positive eigenvalues for the modularity matrix. This allows the algorithm to stop once the network is indivisible, while also preventing the need for determining the size of the subgroups ahead of time.

3.4 Walktrap

In 2005, Pascal Pons and Matthieu Latapy introduced an entirely different method for finding community structure in networks. Their method, named Walktrap, is based on repeated random walks through the graph. The method is motivated by the intuition that a random walk through a densely connected region of the graph will tend to stay within that region [4].

This algorithm makes several assumptions about the data, including that the graph is connected, and that there is an edge between every node and itself. The algorithm begins by dividing the entire graph of n verticies into n communities of one node each. These communities are then iteratively merged based on a distance measure r. The authors define the distance between two communities, r_{ij}, based on the probability of encountering node j given a random walk from i of length t, P_{ij}^t. It is important to note that for a sufficiently high value of t, the probability of ariving at node j approaches $\frac{k_j}{2m}$ where k_j is the degree of node j and m is the number of edges in the graph. In other words, the probability of arriving

at a node depends only on the degree of that node, and not the starting node for the random walk. The authors formally define distance by the equation:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}} \qquad (4)$$

This measure compares the probability of arriving at the same node k when starting from both nodes i and j.

4   Evaluation

To examine the usefulness of these algorithms, I will use them to detect communities in Stack Overflow data. I will compare the performance of my implementations against existing implementations in the igraph package. This will provide insight into the correctness of my implementations as the results should be very similar when compared against igraph. The Stack Overflow data is contains a table of questions, a table of answers to those questions, and a list of tags associated with each question. In the questions and answers tables, there are also fields indicating the user who posted the question. In this data, community structure exists in the form of questions that are about related topics. For example, we would expect a user who is active in the C++ community to answer many questions about C++. It is also possible for users to be active in multiple different communities.

Initially I thought it would be useful to represent both users and questions in the graph we are mining for communities, however, this proved to not be useful. Representing both users and questions forms a bipartite graph where all edges go between user and question nodes. Although this better represents the original data, this makes it much harder to do community detection. For example, if a user is active in multiple communities, there will be more connections joining these communities and it will be harder to find the community structure.

To account for this, I manipulated the graph to better encode the tag information. To do this I removed the user nodes, and created edges between questions that share a tag. It can be seen that this will significantly increase the connectedness of the graph, but this connectedness is useful for algorithms such as Walktrap.

5   Conclusion

References

[1] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[2] M. E. J. Newman. Detecting community structure in networks. *The European Physical Journal B*, 38(2):321–330, Mar 2004.

[3] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[4] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In pInar Yolum, Tunga Güngör, Fikret Gürgen, and Can Özturan, editors, *Computer and Information Sciences - ISCIS 2005*, pages 284–293, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[5] A. Pothen, H. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[6] Alex Thompson. Parallel naratives, December 2016.