CS 224W Final Project Report
Detecting and Analyzing Political Communities in Politics Sub-Reddit Comments

Julien Kawawa-Beaudan, Charles Akin-David, Ikechi Akujobi

December 12, 2016

1 Introduction

Reddit is a popular social media website, which allows users to share stories, photos, news, and opinions. Unlike other media sites, Reddit is naturally divided into "subreddits", a forum dedicated to a specific topic. This allows users to post and browse specific subjects, such as movies, politics, or sports. As frequent users of Reddit, we were interested in examining the community structure of Reddit–seeing whether users naturally fell into categories based on their interests or, in our case, positions on different political topics. We noticed Reddit's subreddits topic structure encouraged people with similar interests to form online communities. The appeal for our project stemmed from the belief that community detection within these online communities could be used for many practical applications such as stopping the spread of computer viruses, detecting spam, or advertising after extrapolating and examining the generated communities.

For our project, we acquired a dataset of comments from 2014 from the politics subreddit. Each entry in the dataset corresponded to a single comment, and had information on when the comment was posted, the author of the comment, the parent comment if the comment was in response to a thread, and the actual text of the comment. In total, the original dataset was 1 GB. We decided to first create a reasonable graph model such that we could evaluate the results of whatever algorithm we ran on the graph. We then ran different community detection algorithms on the model and analyzed the formed communities by running our own algorithms for community organization.

2 Related Work

Community detection is a problem which has been researched extensively. One of the interesting aspects of community detection is the wide-range of metrics and algorithms that can be used to detect communities.

One of the original algorithms for community detection was proposed in 2002 by Girvan and Newman. The main assumption of the their algorithm is that edges with high-betweenness - meaning that these edges occur on the shortest path between pairs of nodes in the graph very often - act as bridges between communities. Their algorithm essentially iteratively calculates the edge with highest betweenness and removes it from the graph until reaching some number of distinct communities.

In practice, this algorithm worked very well on small graphs, however, it requires recalculating the betweenness of the edges at each time-step. This equates to a time complexity of $O(n^3)$, which is infeasible for any of our graph models. In 2004, Clauset et. al devised an optimization to the original GNM algorithm which made run efficiently on large graphs. They optimized based on the property of *modularity*, which is the measure of the effect of a proposed division in a network. The algorithm works by starting each node in the graph as a one-node community, then calculates the change in modularity produced by joining each pair of communities. Then it joins the corresponding communities into one community and recalculates the change in modularity by joining those pairs of communities. The previous step is then repeated until one final community remains.

A very different class of algorithms for community detection are spectral algorithms. These algorithms rely on certain properties about the matrix representation of graphs. First, let us define

the normalized cut of a partitioning of the graph as $Ncut(A,B) = \frac{cut(A,B)}{assoc(B,V)} + \frac{cut(A,B)}{assoc(B,V}$. Here, $cut(A,B)$ is the number of edges crossing between the partitions, and $assoc(B,V$ is the sum of the degrees of the nodes in the A. Intuitively, a good partition should have a low normalized cut value, since there should be few edges between the partitions, and both partitions should be relatively large.

It can be proved that taking the second smallest eigenvector of the Laplacian matrix of a graph gives a partitioning which approximately minimizes the normalized cut. This time complexity for the algorithm is $O(mn) + O(mM(n))$ where m is the maximum number of matrix-vector computations required and $M(n)$ is the cost of a matrix-vector computation of $\mathbf{Ax}$, where $\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$.

The proof is omitted since it appeared on the last homework. However this fact by itself does not give lead to an efficient algorithm for community detection, since the algorithm only applies to bipartitions.

Several researchers have proposed methods for generalizing spectral algorithms to more than two communities. In 1999, Shi and Malik proposed two algorithms based on the spectral methods for community detection. The first essentially uses the algorithm for bipartitioning using the second-smallest eigenvector recursively. Similarly to CNM, this algorithm terminates when splitting an existing partition would increase the k-way normalized cut (which is an generalization of the normalized cut to k-partitions).

The second solution they proposed is a simultaneous k-way cut using multiple eigenvectors. Instead of using the second-smallest eigenvector to compute a partition, the algorithm takes n of the smallest eigenvectors, and runs k-means clustering on the points in an n-dimensional space. Then, they merge the k clusters iteratively until the k-way normalized cut cannot increase by merging a cluster. Finally, they perform the recursive algorithm described above to on the k' clusters. However, they mentioned in their paper that they did not implement this algorithm.

Finally, another algorithm proposed by Ng, Jordan, and Weiss in 2001 is very similar to one described by Shi and Malik. Just like above, they use the smallest n eigenvectors to embed the points into a n-dimensional. They then renormalize each dimension so that the maximum coordinate equals 1, and run k means clustering on the resulting points to find k communities.

3 Model

As stated above, our dataset consists of the comments from 2014 from the politics sub-reddit. Each line of the dataset corresponds to one comment and contains the comment's ID, the parent comment's ID, the author of the comment, the karma score, and the content of the comment itself, minus any links. One of the main challenges of working with the politics subreddit dataset was the size of the corresponding network. The actual dataset is 1 GB. However, this included the actual comment text, which we did not include in our graph structure. We instead brought back this content when we were analyzing how well each community detection algorithm performed. For our model, we chose to represent the dataset as a undirected graph, where an edge between two authors i, j exists if there is some interaction between i and j, such as if i replied to j or vice versa. Furthermore, our algorithms required that the graph they are run on contains only one connected components, since they might erroneously be calculated to be part of the same community even though they should logically be in different communities. As a result, we consider the base graph we use to be the large weakly connected component of the full sub-reddit graph, which contains over 90% of the authors. Table 1 displays information for the dataset.

One major difficulty is our dataset is that the original posts for threads were not included in the dataset. As a result, authors that should have been linked to other authors in our graph through a comment parent were orphaned. This resulted in over 30,000 authors not being repesented in any way in our edge graph. To rectify this, and include a larger number of these nodes in the graph, there also exists and edge between authors who replied to a thread's original post, since the comment correpsonding to the original post for any thread was not included in the dataset.

For comparisons of algorithms used and the resulting communities, we used two different subsets of the overall graph, and ran our communitiy detection algorithms. This was done because of our inability to run our implemented algorithms on our full graph in a timely manner due to the size of the graph and slow runtime of our algorithms. These subsets were created by taking the subreddit graph, randomly sampling a number of authors from the graph (15,000 authors for the smaller

	Full Graph	WCC v1	WCC v2	Small Sub-graph	Large Sub-graph
Nodes	161,715	132729	151,026	4,198	22,978
Edges	1,091,581	933,222	1,090,319	10,325	73,510

Table 1: Model statistics for the dataset and various subsets

subset and 40,000 nodes for the larger subset), getting any edges between the sampled authors in the original graph, and getting the largest weakly connected component of this subgraph. Table 1 provides graph data on the sub-graphs we used for comparing and analyzing our chosen algorithms.

4 Algorithms and Methods

4.1 Modularity Optimization using CNM

For an initial analysis of our dataset, we implemented our own version of the algorithm detailed by Clausel et al. Since modularity is defined as a measure of the quality of a network's division into communities, the division that results in the highest modularity will yield quality communities based on the graph structure. This algorithm works to optimize the well known Girvan-Newman method by building several optimizations on top of the Girvan-Newman method like keeping a matrix noting the change in modularity cause by joining communities i and j.

The algorithm is as follows:

1. Start off with each node in the graph being its own community
2. Initialize matrix ΔQ such that $\Delta Q_{i,j} = \begin{cases} 1/2m - k_i k_j/(2m)^2 & (i,j) \in E \\ 0 & otherwise \end{cases}$
3. Initialize array a such that $a_i = k_i/2m$
4. Get the maximum $\Delta Q_{i,j}$ and join communities i and j
5. Update ΔQ as described below, and update a such that $a_i = a_i + a_j$ and $a_j = 0$
6. Repeat steps 4 and 5 until either one community remains or until the maximum $\Delta Q_{ij} \leq 0$

For step 5, if we delete row j and column j from ΔQ, then only the values in row i and column i need to be updated, since other changes in modularity would not be affected by joining the two communities. For a community c, we update ΔQ_{ic} as follows:

- If c is connected to both i and j, $\Delta Q_{ic} = \Delta Q_{ik} + \Delta Q_{jk}$
- If c is only connected to i, $\Delta Q_{ic} = \Delta Q_{ic} - 2a_j a_k$
- If c is only connected to j, $\Delta Q_{ic} = \Delta Q_{jc} - 2a_i a_k$

Clausel et al.'s CNM algorithm runs in $O(nlog^2 n)$ using a bounded binary tree to store a heap for each row of the matrix such that finding the index of the max Q_k value is easy. However, in our implementation, we used a 2D matrix to represent ΔQ. Because of this, we reinitialize the updated matrix after joining two communities by iterating through the amount of total communities left in a double for loop. Without knowing initially, this data structure change made the time complexity our version of the CNM algorithm $O(n^3)$.

4.2 Spectral Methods

The other class of methods that we implemented were spectral algorithms. Spectral algorithms detect communities by applying linear algebraic methods to the matrix representation of graphs. In particular, we implemented the algorithm for detecting communities based on k means clustering, suggested by Ng, Jordan, and Weiss in 2001.

Both of these methods were covered in the last homework, but as a review, the normalized cut is defined as $Ncut(A,B) = \frac{cut(A,B)}{assoc(A)} + \frac{cut(A,B)}{assoc(B)}$, where $cut(A,B)$ is the number of edges crossing the partitions, and $assoc(A)$ is the sum of the degrees of the nodes in partition A. A small normalized cut corresponds to a partitioning where there are relatively few edges between the partitions and both partitions are reasonable large. Unfortunately, computing a partion which minimizes this quantity is NP complete. However, it can be shown that the eigenvector associated with the second-smallest eigenvalue of L, the Laplacian matrix, maximizes the normalized cut if nodes can be assigned on a range between -1 and 1. Finally, given the second-smallest eigenvector, we can approximate the partition which minimizes the normalized cut by assigning each node to a cluster based on whether the entry in the second-smallest eigenvector is less than or greater than 0. (Fielder, 1973).

4.2.1 Generalizing to Multiple Clusters

Unfortunately, the algorithms described only apply to the bipartition case. In many real problems, including ours, it is necessary to detect more than two communities. So, as suggested by Ng, Jordan, and Weiss, instead of considering the second-smallest eigenvector, we consider n of the smallest eigenvectors (excluding the smallest eigenvector). Intuitively, these vectors are interesting because they attempt to cluster together nodes which share edges in the graph, while maintaining the property that the assignment is orthogonal to the smallest eigenvector. So, we then use these n eigenvectors to embed our data into an n dimensional space and run k-means clustering on this space. Although this approach does not make any guarantees about minimizing the normalized cut, in practice this algorithm performs well on sparse graphs.

Another version of a spectral community detection algorithm involves the modularity matrix, which is defined as $B = A - \frac{dd^T}{2m}$, where A is the adjacency matrix of the graph, m is the number of edges in the graph, and d is a vector of the degrees of the nodes in the graph. As proved in the last homework, the largest eigenvector of this modularity matrix maximizes the modularity of the partition, where modularity is defined as $Q(y) = \frac{1}{4m} \sum_{1 \leq i,j \leq n} \left[A_{ij} - \frac{d_i d_j}{2m} \right] I_{y_i = y_j}$. Although this algorithm is intended for bipartitions, by applying the same k-means clustering technique, we generalized to detecting more communities.

5 Results

5.1 Evaluation

One of the main challenges for this project was finding a good metric to evaluate our communities. In particular, our goal was to detect communities which made intuitive sense, rather than maximized some graphical property such as modularity or the normalized cut. Although modularity and the normalized cut provide good heuristics for detecting communities, simply evaluating communities based on this metric would not have provided farther validation about whether a community made sense.

Intuitively, we expected that reasonable communities should be communities within the population of contributors to the /r/politics subreddit who discuss the same topic. So, using this idea of topic similarity as a guide, we decided to evaluate communities using the cosine similarity of term-frequency inverse-document frequency.

Term-frequency inverse-document frequency, or TF-IDF, is a common metric used for evaluating large datasets of text. The TF-IDF score of a word is the number of times the word occurs in a document, divided by the logarithm of $\frac{N}{1+D}$, where N is the total number of documents and D is the number of documents contains the word. Intuitively, a word which occurs in many documents is less meaningful, or less distinctive in the corpus, and will have a low TF-IDF score, while more rare terms will have a higher TF-IDF score.

Our evaluation metric considers each authors comments as a separate document, and calculates the TF-IDF scores for each author. This gives a large sparse matrix, where each row is a vector containing the TF-IDF scores of the words used by an author. We then calculated the average cosine similarity of the TF-IDF vectors for each pair of authors in the graph. The cosine similarity is defined as $\frac{A\dot{B}}{||A|| ||B||}$, where A and B are vectors. In our case, since every value in the TF-IDF vector is non-negative, the cosine similarity is bounded between 0 and 1, where a score of 1 means

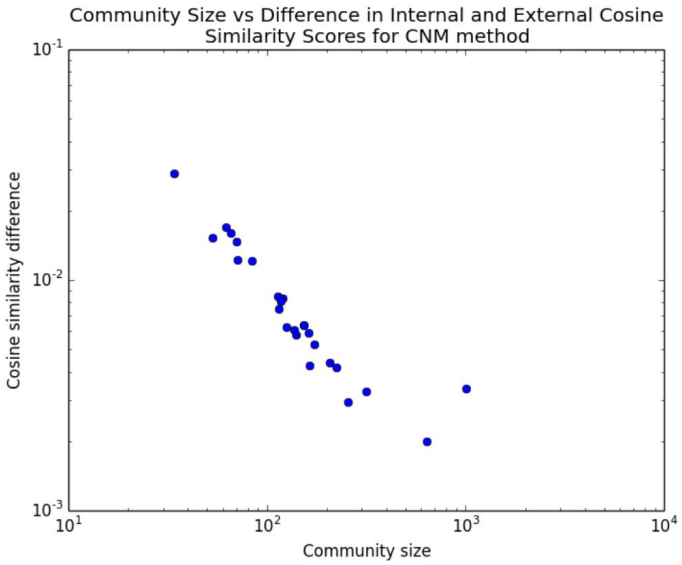


Figure 1: Difference between internal and external average cosine similarity on clusters produced by CNM

two documents are identical. Intuitively, we expect authors within a community to have high TF-IDF similarity, and authors in different communities to have low TF-IDF similarity. So, finally, we calculated the average cosine similarity of TF-IDF vectors, for pairs of authors in the same community and for pairs of authors in different communities.

5.2 Discussion

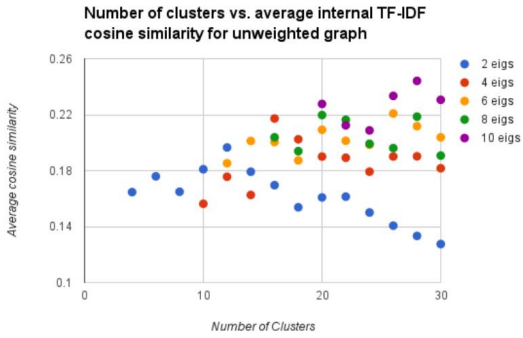
5.2.1 Limitations

A major limitation of the dataset was the lack of original post content for threads that had the same parent topic id. As well as the lack of any concrete labels for users in terms of their interests, party alignments, political beliefs(liberal vs. conservative), etc... One major challenge of this project was determining the general issue comments and threads were discussing using only the content of the comments on that thread. On reddit, the topic of a thread is generally outlined in the original post. The absence of the original posts also orphaned thousands of authors from our graph model, since their only interaction with our network was replying to a original post. The lack of labels made it tough for us evaluate the found communities by each of our algorithms. Due to this, we had to devise and try different types of metrics/scoring techniques to see how well our algorithms formed communities.

At a more high-level, one big limitation of this project is the lack of a clear objective function to maximize. As mentioned in the results section, we decided to compare authors by calculating their TF-IDF vectors, and then computing the cosine similarity of these vectors. This whole metric relies assumption that good communities will have high TF-IDF similarity. However, in some sense we expect TF-IDF to be a poor measure of similarity, because the main topics which unite a community may be very frequent, and have a low TF-IDF score. For example, imagine some community contains users who discuss immigration, which is a well-defined political issue. However, since the word immigration appears very frequently in the dataset and is not used just be users in this community, this community would get a low similarity score because the weight of the word "immigration" in the TF-IDF vector is low.

In addition, this project made assumptions about what a good community should look like - in particular, that the community have a certain theme or interest. Although this community structure is very plausible, it is possible that community detection algorithms would detect some other underlying structure in the graph that it would then focus on when creating the communities.

Results of running k-means variation of the maximizing modularity algorithm.



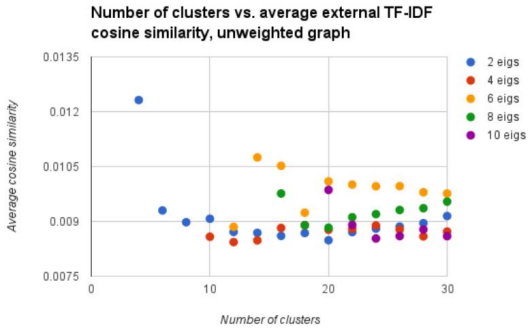
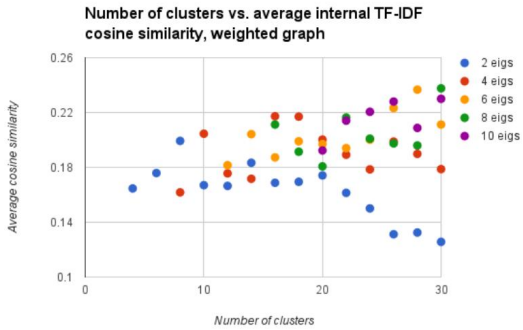


Figure 2: Internal and external average cosine similarity, unweighted graph



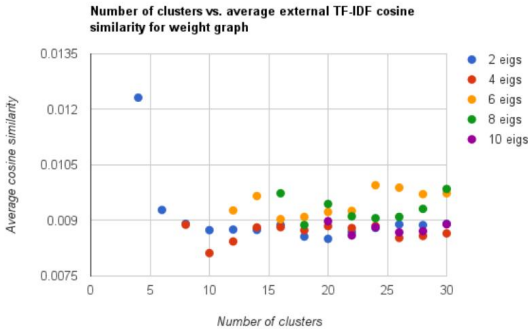
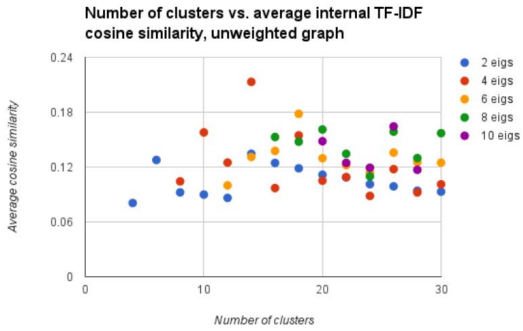


Figure 3: Internal and external average cosine similarity, weighted graph

Results of running k-means variation of normalized-cut minimization algorithm.



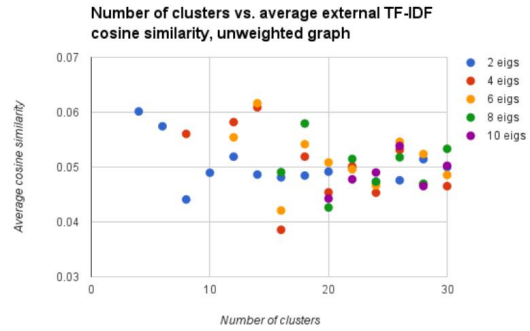
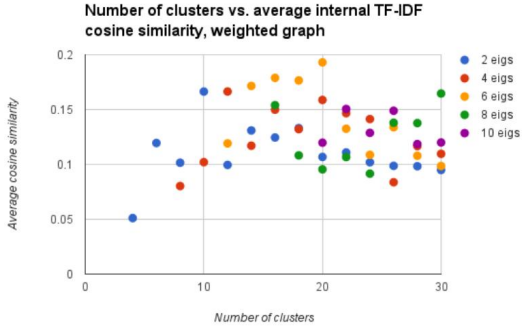


Figure 4: Internal and external average cosine similarity, unweighted graph



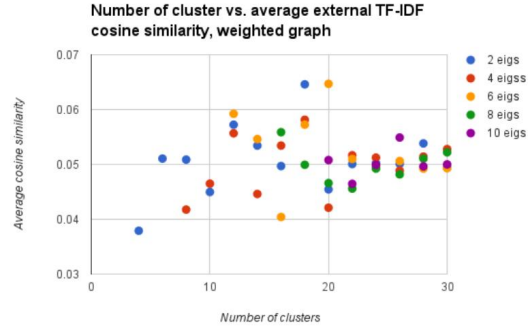


Figure 5: Internal and external average cosine similarity, weighted graph

5.2.2 Charts Analysis

In Figure 1, the difference ($Internal - External$) between the internal and external average cosine similarity by cluster size is plotted. Notice that as the size of the communities increases as the difference measure between their internal and external cosine similarities decreases. The high difference score for small communities have a high internal cosine similarity since the nodes in these communities will be closer together to each other and a low external similarity since these nodes will be further away from all other nodes in the graph. For large communities we observe the opposite effect where we notice high external similarities and low internal similarities. Since these results logical make sense, we know the CNM implementation was correct in terms of finding some underlying structure to use to form communities from our graph. We filtered our communities to not include any single node communities since their internal cosine similarities would be 1.

Figures 2 and 3 show the results from k-means variation of the maximizing modularity algorithm. In Figure 2, we display two charts the first comparing the internal cosine similarities and the second comparing the external both on an unweighted model of our graph. Both graphs show how the cosine score varies with the chosen number of eigenvectors representing the data dimensions. We notice that initializing with 10 eigenvectors usually lead to higher internal cosine similarity scores and lower external cosine similarity scores, which confirmed our belief that given the data more space to live in would give us more information to differentiating groups. Also we noticed that, in general, more clusters lead to better internal cosine similarity scores, while the external cosine similarity usually stayed constant after reaching about 20 clusters. In Figure 3, we display two charts the first comparing the internal cosine similarities and the second comparing the external both on a weighted model of our graph. Here we notice the same effect as with the unweighted graph. It was interesting to see that the weighted graph did not change the results very much.

Figures 4 and 5 show the results from k-means variation of normalized-cut minimization algorithm. Figure 4 focuses on the unweighted graph while 5 focuses on the weighted graph. In Figure 4, we notice the highest internal cosine similarities scores come from initializing with 4-8 eigenvectors and using 15-20 clusters. Also, for external cosine similarities we notice the same initializations lead to the best results. In Figure 5, the globally best initialization for internal cosine similarities was 6 eigenvectors while have 10-20 clusters. For external cosine similarities the best results came from initializing with 2-6 eigenvectors and using 10-20 clusters.

In general the external cosine similarity is a lot lower for normalized min-cut than for max-modularity suggesting that the communities are better formed using the maximizing modularity variation of kmeans.

6 Future work

There are many possible directions that we would have liked to explore farther. One of our original goals for this project was to detect communities with different political opinions. However, we realized many threads consist of users with conflicting political views debating each other. Since our graph structure contains edges for each direct interaction, it seems likely that people with conflicting views would be clustered together, instead of separately. So, instead we decided that our community detection algorithms would probably return users grouped by similar interest, which motivated using TF-IDF to measure content similarity.

However, it is certainly possible that there are metrics other than TF-IDF which are better suited for this problem. In particular, if we actually had the content of the parent topic that started the threads, we could have looked into that content to find the overall topic the commentators were discussing and ran sentiment analysis on the individual comments to see whether or not they were for or against the rooted topic. This way we would have used community detection to see what users were talking about what then utilized sentiment analysis to form the political views set of communities by seeing which topics each user was for and which topics each user was against.

Given more time, it would also have been very interesting to create a different graph which assigns signed weights to edges, representing whether the comments agreed or disagreed with each other. It would be very interesting to see whether the communities detected in this graph correspond to political opinions, as we originally expected. However, out-of-the-box tools for analyzing whether comments agree or disagree do not currently exist, so this would require hand-labeling the dataset.

Finally, there are many other algorithms which would have been interesting to experiment with applying to the graph. One of the algorithms described by Shi and Malik, based on recursively bi-partitioning communities, would have been particularly interesting because it seems like it would avoid the problem of creating very small communities. In addition, other algorithms such as clique-percolation method, mentioned in lecture, would have been allowed for detecting overlapping communities.

7 Conclusion

For this project, we worked on community detection on the dataset of comments from the politics subreddit included finding a graph based on our dataset, running, implementing different community detection algorithms, and coming up with an evaluation framework based on the comments. This is still an open problem, in part due to multiple methods of finding communities and more robust ways of constructing models and evaluating communities using the content of text comments.

8 Work breakdown

Julien: Final report, normalized cut minimization algorithm, TF-IDF evaluation
Charles: Final report, modularity maximization algorithm
Ikechi: Final report, CMN algorithm, sub-graph generation

9 Works Cited

1. A. Clauset, M.E.J. Newman, C. Moore. Finding community structure in very large networks. Phys. Rev. E 70, 066111, 2004
2. A. Ng, M. Jordan, Y. Weiss. On spectral clustering: Analysis and an algorithm. NIPS, 2001.
3. J. Shi, J. Malik. Normalized Cuts and Image Segmentation. IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 22, no. 8, 2000.
4. M. Fiedler. Algebraic connectivity of graphs. Czechoslovak Mathematical Journal, 1973.
5. M. Girvan and M.E.J. Newman. Community structure in social and biological networks. Proc. Natl. Acad. Sci. 99, 8271-8276, 2002.
6. M.E.J. Newman, M. Girvan. Finding and evaluating community structure in networks. Phys. Rev. E 69, 026113, 2004.
7. Roughgarden, T. and Valiant, G. Lecture notes from CS168: The Modern Algorithmic Toolbox, Spectral Graph Theory. May 2016. http://web.stanford.edu/class/cs168/l/l11.pdf
8. Zongqing Lu, Yonggang Wen, and Guohong Cao. Community Detection in Weighted Networks: Algorithms and Applications. IEEE International Conference on Pervasive Computing and Communications (PerCom), 2013.