COMMUNITY DETECTION IN THE TWITTER SOCIAL NETWORK.

NIKHIL CHADDA, ANUPAM PRAKASH, ACHAL SONI

1. Introduction

The Twitter social network connects over 500 million users and generates more than 350 million tweets every day. Organizing the information generated and making it readily accessible is a challenging task, the currently available solutions are the search api and the personalized recommender system. The search api retrieves recent tweets containing the term queried while the recommender system suggests people to follow based on a user's current friends.

The existing apis are not useful for finding user communities interested in a given topic, or for finding the the most relevant users for a given topic. We explore the possibility of building a topic specific user recommender system by combining community information obtained from network analysis with content information extracted from tweets. We identify communities in a subgraph of the twitter network consisting of 115000 influential twitter users and validate the obtained clusters using rare words extracted from user tweets. The computation was carried out on a single machine, however the algorithms are simple and can be scaled up to process larger networks on a cluster.

The project is an attempt to denonstrate that integrating community detection algorithms on the twitter network with tweet content information provides a useful and scalable solution to the topic specific user recommendation problem. The report is organized as follows: section 1 discusses the steps involved in clustering influential twitter users, section 2 discusses approaches for validating the obtained clusters.

1.1. **Datasets:**

We used the following publicly available datasets from the Infochimps platform for our experiments:

(1) A snapshot of the Twitter social network from 2009 containing 42 million users and 1.2 billion edges.
(2) A list of Twitter users ranked according to Trustrank and follower credibility scores, the trust rank is a measure of user credibility based on link and content analysis, the algorithm used for computing the trust rank has not been made public.
(3) A scrape of the infochimps word-bag api that returns the top 100 rare words ranked by $tf\text{-}idf$ scores for a given user.

The trust rank dataset is used to identify a kernel of influential twitter users, we run community detection algorithms for the kernel and validate the clusters obtained using the rare words data set. The trust rank and word bag data sets are for the year 2012 while the network data set is a snapshot from 2009.

The algorithm for computing the trust rank is not publicly available, to validate the trust rank scores we computed the $100,000$ nodes with the highest pagerank for the Twitter network using Graphchi [3] and found that the list had about 27000 nodes in common with the list of $100,000$ users with the highest trust rank. The small intersection size indicates that the trust rank combines tweet content information with pagerank scores and also that the trust rank data set is for the year 2012.

1.2. **The kernel:** We model the twitter social network as an unbalanced bipartite graph with two partitions corresponding to a small kernel of influential users and a large community of users following the kernel users. The influential kernel users are identified as the users having a trust-rank greater than 1.6, figure 1(a) shows a plot of twitter users classified according to the trust rank and follower credibility scores, users lying to the right of the red line are included in the kernel.

Figure 1(b) shows the average in-degrees for the partition of the Twitter graph into a kernel and auxiliary communities. The average in-degrees indicate that the kernel is densely connected and on average each kernel user has many followers from the auxiliary communities. The average in-degree for the users in the auxiliary communities is low and it is a reasonable approximation to ignore the edges coming into the auxiliary community.

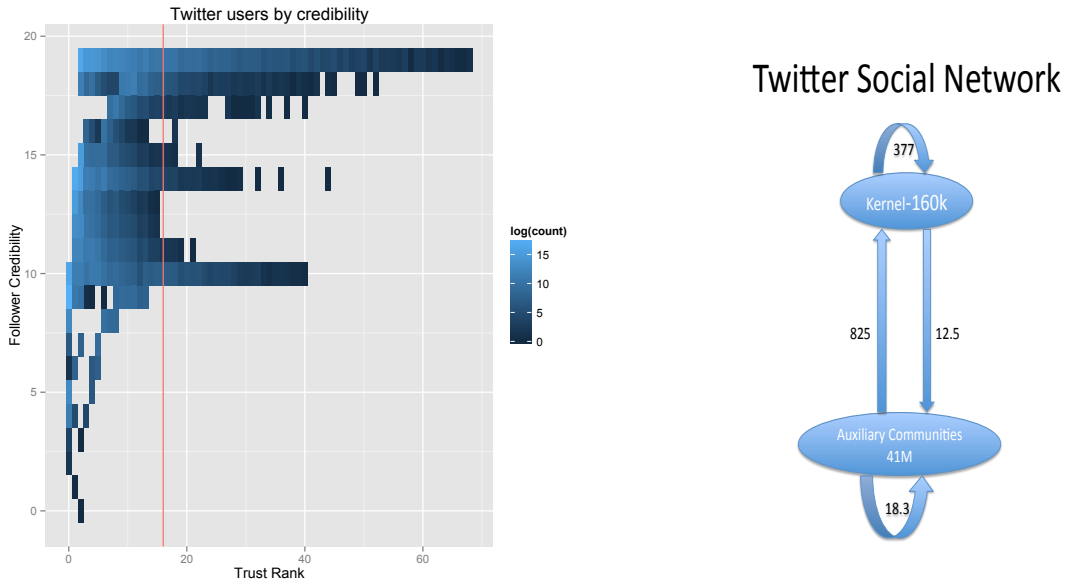

FIGURE 1. (a) Partition of network into kernel and auxiliary community based on Trustrank. (b) Modeling the twitter network as an unbalanced bipartite graph.

The paper [9] suggests that the information propagation network can be modeled as having three layers, a top layer of content producing the kernel users, an intermediate layer of media savvy opinion leaders and content consumers. In figure 1(a), the intermediate layer of opinion leaders are the users with very high follower credibility (20) and intermediate trust rank (in the range $[1, 1.4]$), they are about two million such users, exploring communities within intermediate layer users is a direction for future work.

1.3. **Egonets:** We define the ego-net of a twitter user to be the subgraph of the twitter network restricted to the user and nodes reachable from the user within two steps. An ego-net can be partitioned into a central k-core of users having degree (sum of the in-degree and out-degree) at least k, and whiskers consisting of peripheral users with at most k connections. Communities are densely connected groups of users, thus communities within an ego net are found inside the k cores and nodes in the whiskers can be discarded.

Our approach to community detection is to find communities in the ego-nets of kernel users, the following preprocessing step is applied to remove the whiskers from a user's ego-net:

(1) Discard user if the out-degree is 0 or more than 5000.
(2) Find t such that the t core of the user's ego-net has at most 10000 nodes.

(3) Preprocess the ego-net discarding the vertices that have degree less than k for $k = \min(t, 200)$.

We chose parameters $10000, 200$ for the preprocessing step as we expect the most meaningful communities in the twitter kernel to have sizes in the range $[100, 5000]$. The parameters are design choices and can be changed easily if needed.

The ego-net size is found to be proportional to the out-degree for users in the kernel, the ego-nets of users with out-degree more than a 5000 is almost the entire kernel, so such users are discarded. The value of k is in the range $[3, 30]$ for most users.

A further preprocessing step could involve decomposing the ego-net into strongly connected components, however we found that the ego-nets of kernel users have a giant strongly connected component and a few isolated vertices, finding strongly connected components might be useful for users outside the kernel. All subsequent algorithms run on the pre-processed ego-nets for users in the kernel of the Twitter graph, where the kernel is selected as in section 1.2.

1.4. **Community detection:** Community detection in social networks is a well studied problem and a number of approaches have been proposed in the literature. The paper [1] evaluates the performance of several community detection algorithms on real world networks and finds that different algorithms yield differently structured communities. The differences are explained by the differences in the notion of community and different objective functions that the algorithms attempt to optimize.

Page-rank and random walk based approaches [2] find clusters having small conductance, greedy heuristics in [8], [4] find clusters having high density, link communities [7] and info-map [6], [5] find hierarchical structure while spectral approaches are find sparse bipartitions. We do not expect ego-nets of twitter users to have hierarchical structure so we selected algorithms that optimize the conductance and the density.

The conductance of a cluster S in a directed graph is defined as follows:

$$\phi(C) = \frac{|\delta S|}{Vol(S)}$$

The boundary δS consists of all edges having exactly one end point in S while $Vol(S)$ is the sum of the in-degree and out-degrees for the vertices in S. Page-rank and random walk based approaches attempt to optimize conductance and have been found to produce communities similar to annotated real world communities for several real world networks [1].

The density of a cluster S is the ratio of the number of edges in S to the maximum possible number of edges:

$$d(S) = \frac{E(S)}{|S|.(|S| - 1)}$$

Note that the maximum possible number of edges is $|S|.(|S| - 1)$ as edges in both directions are allowed. Heuristics improving upon the greedy algorithm proposed in [8], [4] find clusters with high density.

For a low conductance cluster, most edges incident to cluster vertices lie inside the cluster while a high density cluster has more edges between cluster vertices compared to the density of the ambient network. We next describe the algorithms from [2] [8] for optimizing the conductance and density, and the modified variants that we implemented.

1.5. **Pagerank based local partitioning:** The page-rank algorithm has two parameters, the reset probability α and the seed distribution s, the algorithm computes the stationary distribution for a process that carries out a step of the random walk on the graph with probability $(1 - \alpha)$ and resets to the seed distribution with probability α. The following pagerank based local partitioning algorithm is algorithm proposed in [2]:

Algorithm 1 Local Partitioning using pagerank

Require: reset probability α, input vertex v

1: Compute page-rank vector π for seed distribution uniform over nodes of G.
2: Compute page-rank vector p for seed distribution concentrated on v.
3: Sort vertices $i \in [n]$ in decreasing order of $p(i)/\pi(i)$.
4: Sweep over vertex ordering obtained and report the cut with the smallest conductance.

The analysis of the algorithm yields the following theoretical guarantee:

Theorem 1.1. *If there is a cluster $S \subset V(G)$ with conductance ϕ and we choose a vertex v from S with probability $Vol(v)/Vol(S)$, then with probability greater than $1/2$, the algorithm finds a cut with conductance $O(\sqrt{\phi log S})$.*

The theoretical analysis requires sampling different seed vectors v and computing page-rank vectors with different reset probabilities, resampling is computationally expensive and not guaranteed to find clusters containing the central node. We implement a modified algorithm that chooses clusters based on the conductance profile for a single execution of algorithm 1 with v equal to the central user.

1.5.1. *Our approach:* We execute algorithm 1 with v equal to the central user and $\alpha = 0.1$ and select clusters based on the conductance profile plot for nodes sorted in decreasing order of $p(i)/\pi(i)$ in step 3 of algorithm 1. Figure 2 shows the typical conductance profile plot for algorithm 1 executed on the ego-net of a twitter user.

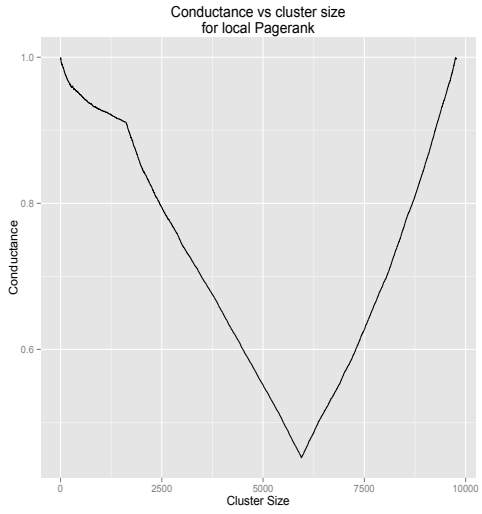

FIGURE 2. Conductance profile for the pagerank based local partitioning algorithm.

Our algorithm identifies points where the conductance profile curve changes direction as clusters, the choice can be motivated as as follows: Nodes with the high values of $p(i)/\pi(i)$ are nodes that a random walk resetting to v is more likely to encounter than a random walk that resetting to a uniformly random vertex from G. If there is a community around the central user, nodes in the community are expected to occur early in the sorted order. When all the nodes in the community have been encountered new nodes in the ordering lie outside the community and the conductance profile exhibits a change in slope.

Algorithm 2 Local Partitioning based on conductance profile

1: Compute sorted order of nodes in step 3 of algorithm 1 with $\alpha = 0.1$ and v equal to the central node.
2: Compute numerical approximation for $\phi''(i) = |2\phi(i) - \phi(i-50) - \phi(i+50)|$ for the second derivative of the conductance profile.
3: Output values where ϕ'' achieves its largest and second largest values (if separated from largest by more than 100) as cluster sizes.

Empirically we observed two points where the conductance profile changes direction for many users. The theoretical guarantee of the theorem 1.1 does not hold for our approach, we validate our approach in section 2 by checking if the clusters discovered correspond to real world communities.

1.6. **Finding high density clusters:** Clusters found using the pagerank based approach are personalized to the user, a different notion of clustering is to find communities having a higher edge density than the ambient network. Dense clique like communities are formalized as 'community kernels' in [8] which are communities for which: (i) No outsider has more friends/followers in the community than any community member, that is if $u \in K$ and $v \notin K$ then $E(u, K) \geq E(v, K)$ and $E(K, u) \geq E(K, v)$.

The high density community kernels were found to correspond to well defined user groups such as politicians, actors and entertainers. The weight balancing $WeBa$ algorithm for finding community kernels was proposed in [8]:

Algorithm 3 Weight balanced algorithm

Require: Cluster size k.
1: Greedy algorithm: Initialize $S = v$, enlarge S to size k by adding the node with maximum neighbors in S, breaking ties arbitrarily.
2: Let S be the output of the greedy algorithm, assign weight 1 to nodes in S and weight 0 to other nodes.
3: Find vertices (u, v) such that $w(N(u)) > w(N(v))$, where $w(N(u))$ is the weight of the neighbors of u.
$$\delta = \min(1 - w(u), w(v), w(N(u)) - w(N(v))/2 \text{ if } u \sim v)$$
Increase $w(u)$ by δ and decrease $w(v)$ by δ.
4: Output nodes having weight 1, repeat $O(n/k)$ times for stability.

1.6.1. *Our approach:* We plotted the density profile for the output sequence of the greedy algorithm (step 1 of Weba) for the ego nets of Twitter users. The density profile plots were remarkably similar across users with the density peaking up at a certain value and then falling off. Figure 3 shows a typical example of a density profile.

We maxima of the density profile was chosen to be the cluster size, and then a weba like heuristic was used to improve the cluster density. The approach found clusters with densities 4-8 times that of the ambient ego-net. Density profiles for some users exhibited maxima for the first 300 nodes in the greedy ordering, these small dense clusters are reported as clique clusters by the algorithm.

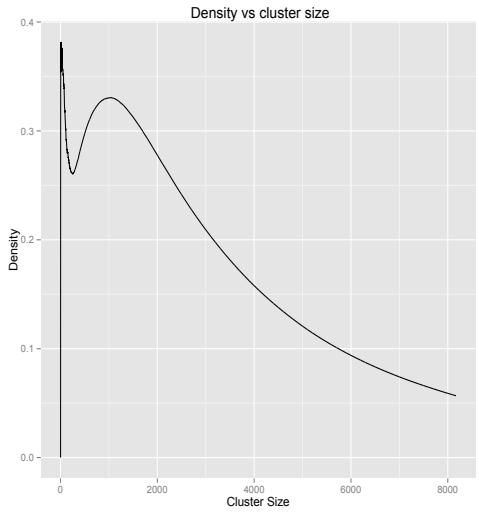

FIGURE 3. Density profile for the greedy algorithm.

Algorithm 4 Dense cluster heuristic

1: Identify the maximum in the density profile of the greedy algorithm. (Report maxima for less than 300 nodes).
2: Assign weights 1 to nodes in the cluster S found in step 1, and 0 to other nodes.
3: Find vertices (u, v) such that $u \notin S, v \in S$ and $w(N(u)) > w(N(v)) + 2$, where $w(N(u))$ is the weight of the neighbors of u. Remove v and add u to S, update weights.

Step 3 ensures that the density of the cluster increases for each iteration, nodes u and v can are greedily to obtain the maximum increase in density. The dense cluster heuristic achieves significant density improvements over the greedy algorithm.

1.7. **Implementation:** Memory and time efficiency are critical to the implementation as ego-nets for typical kernel users have a $5 - 20$ million edges. Our current C++ implementation takes a twitter user id as input and outputs the clusters found in the user's ego-net using algorithms 2 and 4. The kernel size is is small so the the entire kernel graph can be stored in memory.

Extracting the ego-net from the kernel and pre-processing, page-rank based local partitioning and finding dense clusters each consume a few seconds, the processing time is around $10 - 15$ seconds for ego-nets with $10 - 20$ million edges. Computing the page-rank and loading the network into memory are the time critical steps for ego-nets with a large number of edges. Ego-nets with a smaller number of edges can be processed faster.

2. Validating Clusters

Validating the clusters obtained to check if they correspond to well defined real world topics is a non trivial task. The Infochimps rare words api provides a list of 100 characteristic words for each user sorted in order of tf-idf scores. We found that finding words with high tf-idf scores does not identify characteristic words due to high prevalence of marketing related words, we instead used the chi squared statistic for validating clusters. The validation algorithm produces a list of characteristic words for the cluster.

Algorithm 5 Chi squared statistic for validating clusters.

Require: List of cluster nodes C, cutoffs $f = 10$ and $\chi = 50$.

1: Compute the number of times each rare word is used by: (i) Users in the cluster. (ii) Users in the entire kernel.
2: Sort words in decreasing order of the chi squared statistic $(E - O)^2/E$ where E and O are the expected and observed frequency of word usage by cluster users.
3: Output words with frequency greater than f and chi squared score greater than χ.

The frequency cutoff f ensures that the output words are significant for a large number of users in the community, while the chi squared cutoff filters out frequently used but non informative words. We present some topically relevant clusters obtained by running the validation algorithm on the clusters obtained by processing the ego-nets of a 100 users selected uniformly a random.

2.1. **Validating clusters for a random user sample:** Some clusters correspond to well defined real world user communities interested in a specific topic, smaller clusters to more well defined groupings, we found several small clusters corresponding to geographical location and occupation. Some larger clusters corresponding to broad real world topics are presented below, the size and algorithm used are indicated:

(1) Education cluster $(500, 4)$: Significant words include admissions, alumni, student, campus, faculty, professor, research, internships.
(2) Parenting cluster $(250, 2)$: Significant words include moms, kids, child, parenting, preschool, gno, blogher.
(3) Fashion cluster $(450, 2)$: Significant words include fashion, nyfw, style, dress, beauty, collection, shoes, chic.
(4) Hawaiian cluster $(550, 4)$: Significant words include Honolulu, aloha, hawaii, waikiki, mahalo and oahu.

Smaller clusters with sized 100-250 correspond to more specific topics, here are some surprising small clusters found in our sample of a 100 random users:

(1) Ruby programmers community $(100, 4)$: Significant words include railsconf, rubyconf, ruby, rails, rspec, clojure, jruby, github, erlang, scala, java, emacs.
(2) Las Vegas cluster $(150, 2)$: Significant words include citycenter, vegas, casino, nightclub, nevada, poker, resort.
(3) Adobe cluster $(100, 2)$: Significant words include coldfusion, adobemax, adobe, actionscript, cs4, flex, illustrator, builder.

In addition we found communities corresponding to Nigerian music, tourism, financial advice, realtors, yoga, fishing, rap music, medicine, Israel, telephony, health, politics, design, internet based business and foreign language users including Germans, French and Indonesians.

The surprising diversity of clusters in a small random sample of the Twitter network illustrates the vibrant nature of the medium and provides a proof of concept that combining link analysis on graphs with content information extracted from tweets can help find topic specific information rich communities on twitter.

2.2. **Discussion:** The project demonstrates that combining link analysis with rare words extracted from tweets identifies interesting clusters in the Twitter network. Directions for future work include improved approaches for validating clusters, building a topic specific community identification system and scaling up to larger sized networks. The project should be viewed as a proof of concept, further work is required before such a system can be deployed in practice.

References

[1] B. Abrahao, S. Soundarajan, J. Hopcroft, and R. Kleinberg. On the separability of structural classes of communities. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 624–632. ACM, 2012.

[2] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 475–486. IEEE, 2006.

[3] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J.M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[4] N. Mishra, R. Schreiber, I. Stanton, and R.E. Tarjan. Finding strongly knit clusters in social networks. *Internet Mathematics*, 5(1-2):155–174, 2008.

[5] M. Rosvall, D. Axelsson, and C.T. Bergstrom. The map equation. *The European Physical Journal-Special Topics*, 178(1):13–23, 2009.

[6] M. Rosvall and C.T. Bergstrom. Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems. *PloS one*, 6(4):e18209, 2011.

[7] M. Szell, R. Lambiotte, and S. Thurner. Multirelational organization of large-scale social networks in an online world. *Proceedings of the National Academy of Sciences*, 107(31):13636–13641, 2010.

[8] L. Wang, T. Lou, J. Tang, and J.E. Hopcroft. Detecting community kernels in large social networks. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 784–793. IEEE, 2011.

[9] S. Wu, J.M. Hofman, W.A. Mason, and D.J. Watts. Who says what to whom on twitter. In *Proceedings of the 20th international conference on World wide web*, pages 705–714. ACM, 2011.

3. Wikipedia based validation

We attempted to build a classifier that upon given some input, would classify the most important topics present in the text. The goal was to eventually use this classifier to classify user personas into topics by running the classifier on their profile descriptions and rare words extracted from their tweets.

Supervised learning based classifiers like Naive Bayes or SVMs were deemed infeasible as twitter api rate limits did not allow us to mine a significant amount of tweets. Additionally, the issues with standard classifiers go beyond just the requirement of labeled training data - the results produced are largely coarse-grained and lack the precision that could be obtained using a knowledge graph based approach. A knowledge graph gives a better sense of granularity of topics for example: The concept SVM is contained in a hierarchy of parent categories: Data mining, Artificial intelligence, Computer Science.

The Wikipedia ontology tree provides a hierarchical categorization of topics, and that was used for the purpose of the classifier. The Wikipedia dump was preprocessed to remove administrative entries and also entries that don't necessarily represent topics. A directed graph was then constructed, the nodes were Wikipedia articles and categories with directed edges linking nodes to categories and categories to their parent categories.

The number of articles are an order of magnitude of order greater than the number of categories, and we discarded several less informative articles. Discarding articles loses the information that tells us that Category X and Category Y are similar if they share many common articles. Edges were added between Category X and Category Y if they shared many entities.

The core idea was to first identify what the major keywords were in a tweet or profile description, which was aided by a POS-tagger from CMU specifically designed for tweets. This preprocessing state of the tweets helped to tokenize tweets that are sometimes hard to otherwise do so, and annotate what part of speech each token was, so meaningless words like prepositions could be thrown out. We then identified which articles or categories had the same title as the tokens in the tweets (we looked at single tokens, bigrams of tokens, tigrams, and quad grams). We then looked up (if the matching entry in Wikipedia was an article) in a database (MongoDB) what parent categories (and hence what nodes in the Wikipedia graph that was kept in memory) were for that topic. If the title of the category matched the token itself, then that was identified as well but there

was no need to look up in a database because it was part of the in-memory graph (the in-memory graph had categories and their links).

After constructing the set of initial categories, we found all paths between these starting categories in the graph up till depth 10. This was in essence a subgraph of the original Wikipedia graph, that contained all the relevant nodes and semantic links between the initial topics that we identified. Before running a ranking algorithm on this graph, we also identified which of these nodes had a matching entry in the WordNet graph. The same subgraph algorithm was the ran on the WordNet graph, with the matching entries from the Wikipedia subgraph and Wordnet subgraph identified as the initial starting points. PageRank was then ran on this subgraph. The reason for doing so was that the observation that the Wikipedia categorization ontology was not as precise and strict as desired, so to disambiguate and down-weight unlikely topics that otherwise might surface in the Wikipedia subgraph, we reinforced it with the WordNet subgraph, which is far more precise and semantically oriented.

After obtaining the PageRank values of the common nodes, PageRank was on the directed subgraph of Wikipedia, with the initial probability distribution having equal distributions amongst all the initial starting nodes and having these probabilities propagate through the graph through the iterations of the PageRank algorithm. The reason for not giving initial probability distributions for intermediate nodes that weren't initially matched with tokens in the input was because we didn't want unrelated topics to continue to reinforce each other and give each other higher values even though they were unrelated to the initial tokens, so the decision was made to let the semantic relations dictate how strong those extraneous categories were.

After the PageRank was ran, a weighted average was taken for the common nodes, and then for each topic, the top parent categories were chosen. Then from these parent categories, the top categories were chosen, and so on so forth. After 4 to 5 iterations of this, we had a sense of the best topics for a token at incremental depths/steps in the Wikipedia graph (the first iteration told us the best immediate topic categories for a given token, which is one dimension of granularity less, and then iteration 2 gave us 2 steps away, and so on so forth). Some intelligent combination of all the classification for the nodes is still to be done.

One advantage of Wikipedia also lied in that topics were grouped together by other names for the topic (i.e.: Obamacare and Patient Protection and Affordable Care Act were treated as the same node). This helped to encapsulate some of the common ways in popular culture to describe the same topics.

There is a lot of work left in the future for this classifier. More relevant semantic analysis needs to be done, to simply figure out which edges are more likely than others, and to build a more precise knowledge graph. More statistical and machine learning work to incorporate real-life Twitter data to help the classifier assign probabilities to paths and edges and nodes would be done as tweets and such were accumulated. This data could also be used to introduce new topics, and to figure out what terms refer to the same thing (CP3 is a nickname for Chris Paul, the point guard of the Los Angeles Clippers). A better way to parse the unconventional structure and terms in tweets also has to be devised. Any approximations that would reduce the computational load would need to be investigated as well.

3.1. **Visualization:** The primary goal for visualizing our results was to validate and compare the clusters that were generated from the variety of different algorithms and approcahes to our problem. Specifically, we ran both the Weight Balanced and Personalized Pagerank algorithms on a number of high profile Twitter users as an initial sanity check on the quality of the algorithms. We were able to visually verify the relative size of each of the generated clusters and their corresponding connections within the user kernel. The visualization structure for each user's ego-net was relatively simple: the generating user was placed at the center of the graph, and a node was added for each

clustering algorithm. Edges were then added between each algorithm node and the users that were placed in the cluster by that algorithm. One can easily imagine this structure being extended as more clustering algorithms are considered or created, providing a standard visual comparison among clustering methods. Additional features could include statistics displayed on screen for each of the clusters and displaying the current structure against a faded backdrop of all users not included by any clustering algorithm. Overall, the visualization features served the purpose of more clearly communicating the complex steps taken to reach our overall project findings and providing an additonal form of basic validation for our clustering algorithms.

Department of Electrical Engineering and Computer Science, University of California, Berkeley.
E-mail address: nchadda, anupam, achalsoni@berkeley.edu