CS224W Project Report

Chong-Kwan Tan

Project: Finding Community Structure on oDesk

The Problem

oDesk is a job site where employers post jobs, and contractors apply and get paid for work done. We are interested in the solving the following questions:

1. Given a job opening, find a list of contractors suitable for the opening.
2. Given a contractor C, find a list of contractors similar to C.

The answer to Q1 lets us recommend contractors to fill an opening requiring a certain skill set (e.g., find me a contractor skilled in python and mysql.) The current algorithm used on oDesk is the obvious one: match the skills required by the job to the skills the contractors claimed they have.

The answer to Q2 enables employers to look for a contractor similar in capability to another they are satisfied with (e.g., find me a contractor similar to Peter, who I am very happy with but is currently unavailable.) oDesk currently do not support this feature.

The Data

We define a node as a contractor, and an edge connects two contractors if they applied to the same opening.

LARGE SET: We are able to obtain records of job applications submitted to oDesk for the first half of 2011. Initial counts shows a total of 90,552 employers created 457,250 openings, on which 272,761 contractors submitted a total of 5,734,033 applications. Joining contractors that applied to the same jobs resulted in a graph of 272,761 nodes and 84,722,454 edges.

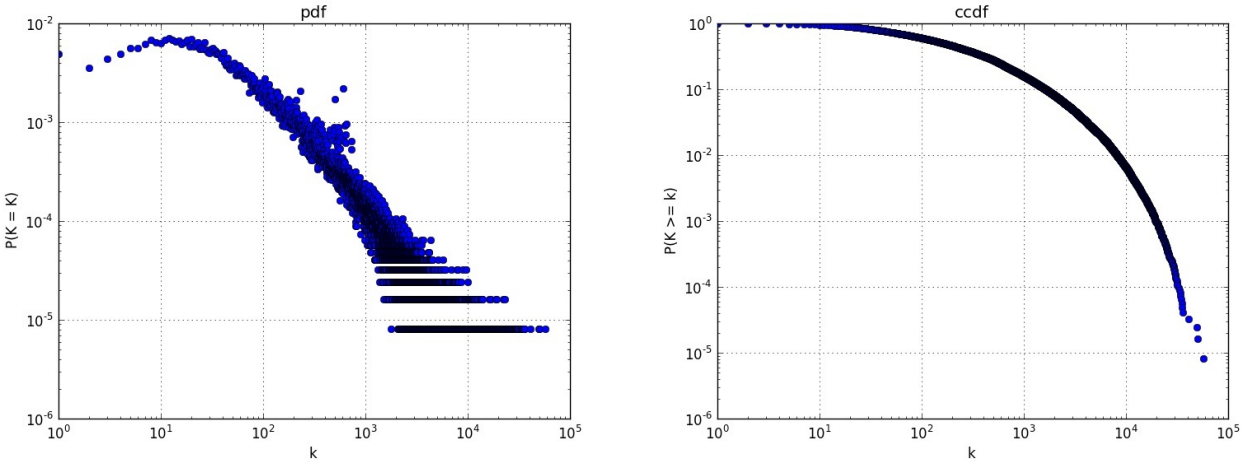
MEDIUM SET: Filtering out openings that did not result in any hiring nets 46,619 employers, 245,960 openings, 134,255 contractors, and 1,409,750 applications. (134,255 nodes and 38,612,154 edges).

SMALL SET: From the Small Set, keep only those in MEDIUM SET that relates to contractors who had worked in more than 3 jobs. This resulted in 42,696 nodes and 4,307,972 edges.

The nodes in the SMALL SET exhibit the following properties (k is the degree of a node):

* Min(k) = 1
* Max(k) = 57,219
* Avg(k) = 678
* Stddev(k) = 1696

Plotting the degree distribution of the nodes:



From the PDF and CCDF, we can see that the distribution is not power-law.

The Method

Girvan [1] introduces the property of community structure (sometimes also called clustering) in which network nodes are joined together in tightly-knit groups between which there are only looser connectors. In contrast to the idea of the betweenness centrality of a vertex, defined as the number of shortest paths of other vertices through the vertex, Girvan defines the *edge betweenness* of an edge as the number of shortest paths between pairs of vertices that run along it. If a network contains communities that are only loosely connected by a few inter-community edges, then all shortest paths between different communities must go along one of these few edges. Girvan describes an algorithm that accentuates and removes these high edge betweenness links to reveal the communities as cluster of nodes become separated from one another. The algorithm incrementally breaks the network down into separate communities all the way down to single vertex. The output of the algorithm is in the form of a dendogram which represents an entire nested hierarchy of possible community division for the network.

Newman [2] details the original algorithm in [1], and further define a measure called *modularity*, meant to quantify the quality of a particular division of a network. As the algorithm in [1] outputs the dendogram of the network, the modularity measure seeks to tell where we should cut the dendogram to obtain a sensible division of the network. At each division in the dendogram, the modularity of the network is calculated. Local peaks of the modularity graph indicate particularly satisfactory splits. The authors showed that the application of the measures produces the expected result when run against networks with known communities, and accurate and reasonable result when run against networks without prior knowledge of communities.

Clauset [3] proposes an algorithm that speeds up the algorithm in [1] that runs in $O(n^3)$ time for a network of n nodes. We coded the algorithm described in [3] to find community structure among the network of contractors in the data sets. The clustering algorithm is of order O(md log n), where m is the number of edges and d is the depth of the dendogram.

We start from the initial state where each node is its own community. The algorithm keeps a sparse matrix that maintains the value of ΔQ for communities i and j, where Q is the modularity of the network. ΔQ is the contribution to Q should we merge communities i and j. Each time we go through the loop, we select the max ΔQ, and merge the two communities corresponding to the ΔQ, and recompute ΔQ for any communities that were affected. Naturally, Q keeps increasing as long as ΔQ is positive, and Q keeps declining as soon as ΔQ becomes negative. We exit the loop at maximum Q, which is the precise moment when ΔQ turns negative. At the end of the loop, we should be left with a set of distinct communities, each with multiple nodes, that cannot be merged without decreasing the modularity of the network.

We wrote the clustering program in the GO language for its speed and its parallel processing capability using coroutines. We have not, however, utilize any parallel processing in the current code, i.e. at this time it is single-threaded. The input of the program is a file containing the edges. The output is a dendogram.

To debug and verify the code, we run it against the Karate Club data. The initial output was not optimal. We went back to look at [2] Newman in detail, and were able to find an error in [3].

From [2], page 16:

[49] As discussed in [33], it is crucial to make sure each edge is counted only once in the matrix e_{ij}—the same edge should not appear both above and below the diagonal. Alternatively, an edge linking communities i and j can be split, half-and-half, between the ij and ji elements, which has the advantage of making the matrix symmetric. Either way, there are a number of factors of 2 in the calculation that must be watched carefully, lest they escape one's attention and make mischief.

As it turned out, the initial calculation of Q in [3] is missing a factor of 2. After the correction, we were able to obtain almost optimal result of the Karate Club data: 3 distinct clusters, where if we let the algorithm run for one more iteration, we would have gotten the correct answer of 2 clusters.

We were reasonable satisfied with the correctness of the algorithm, and we proceeded to optimize it for large data sets. Efforts in parallelizing the algorithm were not fruitful. It turns out that each step of the algorithm does only very small amount of computation, and our effort in using coroutines resulted in more overhead than savings. We then focused our attention on optimizing the data structure involved in the algorithm, and were able to speed it up by a factor 4.

We were only able to run our algorithm on the SMALL SET due to time constraint.

Evaluation

To measure correctness of the communities, we match the output of the clustering algorithm against the contractor self-described category. The table below shows the top-8 matching ratio of all contractors which is highly encouraging.

cluster	ctype1	cnt1	ctype2	cnt2	cnt_total	ratio
814823	Web Development	9369	Design & Multimedia	999	11595	0.89
1331543	Writing & Translation	2526	Administrative Support	1688	6981	0.60
29897	Writing & Translation	3214	Web Development	1345	5876	0.78
395099	Writing & Translation	194	Sales & Marketing	75	453	0.59
1026009	Writing & Translation	21	Administrative Support	16	53	0.70
856396	Writing & Translation	10	Design & Multimedia	8	21	0.86
717037	Web Development	7	Writing & Translation	10	19	0.89
296857	Writing & Translation	8	Administrative Support	7	17	0.88

Conclusion

To answer Q1, we simply find the community that matches the required skill set specified by the opening, and return the nodes in the community.

To answer Q2, we first pinpoint the node representing the sample contractor C, and return the other members of C’s community.

The algorithm to pick a contractor that would result in a positive outcome is a separate problem, involving:

1. the feedbacks of the contractor
2. how busy is the contractor
3. is the contractor looking for work
4. does the contractor reside in the same geographical region of other hires in the employer

Central to our project is the empirical proof that community structure built using the behavior of the contractors is as good or better than the self-classification of the contractors. We are highly encouraged by the result, pending more detailed analysis using large data set.

References

[1] M. Girvan and M.E.J. Newman. *Community structure in social and biological networks.* Proc. Natl. Acad. Sci. 99, 8271-8276, 2002.

[2] M.E.J. Newman, M. Girvan. *Finding and evaluating community structure in networks.* Phys. Rev. E 69, 026113, 2004.

[3] A. Clauset, M.E.J. Newman, C. Moore. Finding community structure in very large networks. Phys. Rev. E 70, 066111, 2004