Refining Queries for Search Engines by Community Mining Approaches

Jiyang Chen

Abstract

In this document, we describe a project to design and implement a novel approach to refine query terms based on community detection methods. Our approach is based on the modularity measure [2] for community structure. We also propose an improvement of the current modularity measure.

1 Introduction

Query refinement for search engines has attracted more and more research attention in recent years. The task of query refinement is to improve the "quality" of queries that user submit to search engines to give better results, where better means closer to what the user tries to find. NLP and machine learning scientists have been applied number of approaches to solve the problem, but their results are not satisfied.

Let's give a simple example of query refinement. "Jaguar" is a common word to search. Usually there are two kinds of people who submit that query: people who are interested in "Jaguar the animal" and people who are interested in "Jaguar the car". However, popular search engines do not provide any assistance to separate the two, pages with different topics are merged in the search results. If the search engine can detect there are two different kinds of jaguars, we can suggest a better query to achieve higher quality result, e.g., Google can display "Do you mean 'Jaguar animal' or 'Jaguar car"' and let user click to choose one. Other example queries include "Michael Jordan" (famous basketball player and U.C. Berkley professor), "saturn" (car and planet and company and video game console), and so on. The problem can be simply stated as follows: *"Given a query term, how to decide whether it has multiple meaning and should be refined? How to refine the query if we know it should be refined?*

In this document, we propose a novel approach to refine queries based on a hierarchical clustering method for community mining and a quantitative measure for community mining evaluation. The basic idea of our approach is to separate the problem into two steps. Specifically, given the questioned query term, we first extract name entities from Google search engine to build the entity relational graph, then discover communities in that relational graph, and finally generate possible refinement for queries.

2 Preliminaries

2.1 Community Mining

Many structured data of scientific interest can be represented as networks, where sets of nodes or vertices joined together in pairs by links or edges. Examples includes social networks such as researcher collaboration and friendship network, the World Wide Web (WWW) such as the web page hyperlink network, and biological networks such as neural networks and food webs. A property that many of these networks have in common is the *community structure* in networks, which means that there exists densely connected groups of vertices, with only sparser connections between groups. *Community Mining*, which focuses on the detection and characterization of such network structure, has received considerable attention over the past few years. A community can be defined as a group of entities that share similar properties or connect to each other via certain relations. Identifying these connections and locating entities in different communities is the main goal of the community mining research.

2.2 The Modularity Measure

Newman and Girvan [2] proposed a metric named modularity measure (Q) to compute the fit between the community division and the network link structure. Consider a particular division of a network in to k communities. Let us define a $k \times k$ symmetric matrix e whose element e_{ij} is the fraction of all edges in the network that link vertices in community i to vertices in community j.

The trace of this matrix $Tr(e) = \sum_i e_{ii}$ gives the fraction of edges in the network that connect vertices in the same community, and obviously a good community division should have a high value of this trace. However, the

trace on its own is not a good indicator of the quality, since placing all vertices in a single community would give the maximal value 1 while giving no information of the community structure at all. Thus, the row sum $a_i = \sum_j e_{ij}$ is defined to represent the fraction of edges that has at least one end in community i. So, a_i^2 is the expected fraction of edges within community i if the edges were distributed randomly. Thus, the modularity measure is defined by:

$$Q = \sum (e_{ii} - a_i^2)$$

This quantity measures the fraction of the edges in the network that connect vertices of the same type, i.e., within-community edges), minus the expected value of the same quantity in a network with the same community division but random connections between the vertices. If the number of with-in community edges is no better than random, $Q = 0$. Values of Q close to 1, which is the maximum, indicates strong community structure. Q typically falls in the range from 0.3 to 0.7 [2], high values are rare.

The modularity measure has drawbacks. The Q value is sensitive to the number of communities and nodes in each community. It cannot compare community quality for different networks or different number of communities for the same network. (For different number of communities, we may compute the average of Q.)

3 Improving Modularity Evaluation

As discussed in the above, given a network of k communities, modularity is defined as $Q = \sum_{i=1}^{k}(e_{ii} - a_i^2)$ where e is a $k \times k$ matrix whose element e_{ij} is the percentage of link that connect nodes in community i with community$j$. a_i is the sum of row i for e, which is the fraction of edges with at least one end in community i, thus a_i^2 is the expected fraction of edges within community i if the edges were distributed randomly. Values of Q close the 1 indicate strong community structure and values close to 0 indicate no community structure. However, the modularity measure does not explicitly take non-links into consideration [3], i.e., it only measures how good the discovered community structure fits the existing links (connected vertices should be in the same community), but fails to measure how good the structure fits the non-existing links (disconnected vertices should not be in the same community). For example, Figure 1 and Figure 2 show two networks with communities. They both have the same Q value, which is 0.216 (since both has 11 edges in community 1, 5 edges in community 2, 2 edges between),

but apparently the second one is a worse division. The reason that Q cannot separate them is because the second figure has more non-links which is not considered by Q measure. Here we proposed an improved measure based on Q, we call it *Max-Min Modularity*.

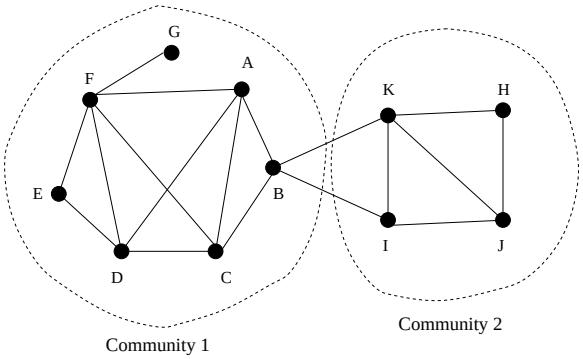

Figure 1: Network Community Example

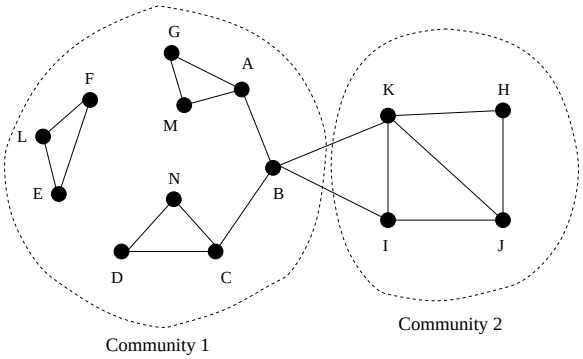

Figure 2: Network Community Example

The idea of Max-Min Modularity is based on the intuition that a good division of a network into communities is not merely one in which the number of edges between groups is smaller than expected, it is also one in which *the number of non-existing edges within groups is smaller than expected*. Only if the numbers of between-group edges and within-group non-existing edges are significantly lower than would be expected purely by chance can we justifiably claim to have found significant community structure. Equivalently, we can examine the number of edges within communities and non-existing

edges between communities and look for divisions of the network in which this number is higher than expected. These two approaches are equivalent since the total number of edges is fixed and any edges that do not lie between communities must necessarily lie inside one of them.

We already defined the Q for edges within communities, with larger values indicating stronger community structure. We note it as Q_{max} to make a difference.

$$\begin{aligned} Q_{max} &= (number\ of\ edges\ within\ communities\ in\ G) \\ &\quad -(expected\ number\ of\ such\ edges\ in\ G)\ [2] \end{aligned}$$

To measure the other part of the measure, we first transform the network of interests G to G':
Given $G = (V, E)$, we create $G' = (V, E')$, if there is no edge between vertex i and j in G, connect them in G'. (if $(i, j) \notin E, (i, j) \in E'$).
We apply the Q function on G' based on the same community structure to get Q_{min}:

$$\begin{aligned} Q_{min} &= (number\ of\ edges\ within\ communities\ in\ G') \\ &\quad -(expected\ number\ of\ such\ edges\ in\ G')\ [2] \end{aligned}$$

The smaller Q_{min} is, the stronger community structure we have on the original network G. Thus the max-min modularity principle requires we minimize Q_{min} while maximize Q_{max} at the same time. All these requirements can be simultaneously satisfied by the following objective function (note that the maximum possible value of Q is 1):

$$Q_{maxmin} = Q_{max} * |Q_{min} - 1|$$

4 Our Approach

Our approach is consist of two steps, as described in the following.

4.1 Generate the entity relation network

At first, we need to extract entities and their relations from the web. There are various methods to do that. Let me speculate one possibility I can think of.

1. Submit the given query word to Google, collect the top k pages returned as query results.

2. For each page, extract the top n entities based on TFIDF score over the whole dataset.

3. Now we have at most $n * k$ entities (there may be some overlap). If entity x and entity y have appeared in the same sentence, we count as weight 1 and accumulate over all documents.

4.2 Refine queries based on communities

After we have build the relational network of keyword entities, we can apply a hierarchical clustering method to find communities based on the improved modularity measure, discussed in Section 3. Our approach is similar to [1]: assume every entity is in its own community at the start, we merge two entity into one community if we find an increase in modularity, otherwise no change. We keep merging until we find no more possible modularity increase.

Ideally, the entity relational graph should be separate to several communities, whose number is equal to the different meaning of the given query term, e.g., we expect two big communities for "Jaguar" and "Michael Jordan". After that, we can select the most related entity to the given query word in each community as the possible query refinement.

5 Conclusion

This document explains a novel idea of refining query terms with the help of community mining methods. This topic attracts a lot of attention in the research field and is still waiting for a satisfactory solution. However, my ideas are still preliminary and there are still a lot of work to be done.

References

[1] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 2004.

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

[3] Jerry Scripps, Pang-Ning Tan, and Abdol-Hossein Esfahanian. Exploration of link structure and community-based node roles in network. In *ICDM*, 2007.