Analyzing Students Logs in Open Online Courses Using SNA Techniques

Abstract

There are a growing number of courses delivered using Massive Open Online Course environments (MOOCs). In these courses, as with any, interaction between online students and learning resources can play in improving the learning environment. These improvements could better accommodate each individual’s needs based on their prior knowledge and skills. Most current MOOCs use a one-size-fits-all model with a singular curriculum. We suggest that a singular curriculum does not provide a particularly good match for this broad spectrum of students with diverse educational backgrounds. This problem is manifest in the relatively low rates of completion in many MOOC courses. As an alternative, we propose the notion of Personalized Open Collaborative Courses (POCCs). This design tracks student behavior through each stage of the course and, based on each student’s performance and prior-knowledge, the course content and delivery is personalized. This study follows the emerging research in Social Network Analysis (SNA) techniques. More specifically, we examine the concept of detecting communities of users within a large course with an emphasis on designing a personalized, social, recommender system. We use three of the most popular community detecting algorithms: Fruchterman Reingold, ForceAtlas2 and Yi Fan Hu for community detection. Finally, we define an algorithm to build dynamic social recommender systems.

Keywords


Introduction

Most current MOOC environments (i.e., Edx, Coursera, Udacity, Udemy, P2PU, etc.) utilize a one-size-fits-all model with a singular curriculum. This is by design as one of the goals of MOOCs is to educate a large potential audience with minimal development and delivery costs. We believe that this singular curriculum does not fit the needs of many of the students – particularly those with diverse educational backgrounds. Evidence about this problem can be noticed in Figure 1, where the persistence rate is so low. Actually, only about 10% of those enrolled students finish a class (Hill 2013). This is a point that many people criticize. One alternative that may help overcome some of the limitations may be the use of Personalized Open Collaborative Courses (POCCs). This new generation of Massive Online Courses has the potential to better meet individual needs by using an adaptive, personalized curriculum. This design tracks a student’s behavior through each stage of the course. As feedback is collected on performance and prior-knowledge, the content of the course will dynamically adapt to better aid student mastery of the concepts. The duration of POCC courses will vary based on student ability and achievement.

Since 2006, Western Kentucky University (WKU) has hosted over 25 Mid-sized Open Online Courses using the HyperManyMedia (HMM) Platform (HyperManyMedia 2006). The HMM learning management system serves as an experimental environment for developing and evaluating machine
learning algorithms. For more details, refer to (Zhuhadar and Nasraoui; Zhuhadar et al.; Zhuhadar et al. 2008; Zhuhadar et al. 2009).

Figure 1.

Much of this work builds upon the growing body of research in Social Network Analysis (SNA) techniques. In particular, we seek to better understand the concept of detecting communities of users in the HMM platform with an emphasis on developing a methodology to design personalized, social, recommender systems. The broad approach involves using the social network layout to visually analyze the HMM’s logfile of user activity. These logs provide rich data and potential insight into the structure of HMM’s communities, i.e. which groups of users are included in these communities. The analysis uses three popular community detecting algorithms, such as, Fruchterman Reingold, ForceAtlas2, and Yi Fan Hu.

In the following section we introduce the background of recommender systems and social network analysis. This is followed with the methodology used to detect communities of learners and the experimental design used. The report concludes with findings and directions for future research.

Background and Related Work

Recommender systems can be traced to various research fields, starting in 1979 with the extensive work of (Rich 1979) in user behavior and cognitive science, referred to as user modeling via stereotypes, followed by the study of the evolution of marketing using forecasting theories, in particular, the contributions of marketing to strategic management (Biggadike 1981); then in 1989, through Salton's work in information retrieval, studying Automatic text processing as “the transformation, analysis, and retrieval of information by computer (McGill and Salton 1983).”

Popular companies started to invest in online storefronts, such as Amazon.com and eBay. However, from the mid-2000s to date, e-commerce portals have been facing a major problem because of the extensive growth of the content (items) provided to customers and the exponential growth of; online users, which is known as the bottleneck problem in online traffic. As a consequence, recommender systems were the natural extension to e-commerce portals. However, our interest in this paper is to provide an overview of recommender systems in academic repositories. We believe that one of the main services that can benefit from the usage of recommender systems are digital libraries and MOOCs.

On one hand, ACM, IEEE Xplore and CiteSeer incorporated some techniques that could be considered as a form of recommendations (with little success). For example, ACM Portal provides two types of recommendations: (1) a content-based research tool known as “find similar articles”. The mechanism used to find similar papers involves three techniques: cluster analysis, dictionary and thesauri. The retrieved documents are ranked based on date, publisher, or relevance, but there is no reference to the type of measure used in the ACM Portal, as cited in (Neumann 2009), 2) behavior-based recommendations presented as “Peer-to-Peer readers of this article also read”.

Twentieth Americas Conference on Information Systems, Savannah, 2014
On the other hand, Melvyl (Larson and Graham 1983) and TechLens (Konstan et al. 2005) are considered as the most successful recommender systems in Academia. Melvyl, which has been implemented by the California digital library uses a simple technique to provide recommendations to users. First, it generates a graph of all the purchased documents in the library, then each document is considered as a weighted node (the weight represents the number of purchases). Therefore, the recommendation for a given document is based on the neighboring nodes (documents), which are sorted according to their edge weights. The second is TechLens, which is specialized for the domain of scientific papers; it uses hybrid recommendations combining a collaborative filtering and a content-based approach. The system uses linked graph theory where each research paper is considered as a node and the citations inside each paper are considered as recommended nodes. Also, the system uses a more complex collaborative filtering (CF) technique that considers each cited paper as an input, therefore all citation papers are considered as recommendations too. This technique is referred to as Dense CF. Finally, the system applies a content-based recommendation technique (TF-IDF) on the list of all the recommended papers. Thus, the most similar papers are recommended to the user.

Our research approach shares some similarity with the above techniques. Details about these similarities are described in the following sections: First, we describe the methodology used to provide the learner with communities based social network analysis recommendations; then, we follow the recent line of research in Social Network Analysis (SNA) techniques in implementing HMM recommender system. SNA techniques are mainly popular in the area of Social Media applications. In our research we map the evolution of SNA into the educational domain. For instance, recently, (Brandt and Leskovec 2014) examined the evolution of five social networking sites: Twitter, Flickr, DevianArt, Delicious, and Yahoo! Answers and they found that friendship and status-oriented-linking-behaviors provide insight into the evolving structures of social networks; whereas, in our study, we used the structure of social networks to provide recommendations to online learners. More specifically, we study the concept of detecting communities of users in HMM’s platform with an emphasis on building an optimal methodology to design a personalized, social, recommender system. We use three of the most popular community detecting algorithms: Fruchterman Reingold, ForceAtlas2 and Yi Fan Hu for community detection. Finally, we defined an algorithm to build our dynamic social recommender system.

Methodology

The primary goal of this research is to develop a method of detecting communities of similar user groups from a large log of user activity and interaction. Force-directed methods were used to discover similarity between users. We experimented with three algorithms: i) Yifan Hu multilevel algorithm (Hu 2005) and, Fruchterman Reingold (Fruchterman and Reingold 1991) algorithm, and (iii) Force Atlas 2 algorithm (Mathieu Jacomy 2011).

To start, we must select the criteria on which we can categorize each community and then look at various options available for each and proceed accordingly. We define the set of various force laws that are to be considered for the categorization of the communities. Once the communities have been recognized, we need to make sure that each one has its own energy state, which determines the relevance level of that particular community in its range of proximity.

Modularity measure

Modularity Measure is used to compare the quality of the partitions obtained by different methods. This method was proposed by (Blondel et al. 2008) and its parameters were used for i) evaluation and ii) suggesting the best force directed method to be used for community detection.
Definitions and notations

Let us assume that \((V_u)\) is a user vertices and \((V_c)\) is a learning resource vertices. As shown in Figure 2, an edge \(E\) is defined from \(V_u\) to \(V_c\) if a user \((V_u)\) visited the learning resource \((V_c)\). Thus a Graph \(G = \{V,E\}\) is defined, where \(V\) is the set of all vertices \(\{V_u, V_c\}\).

We used:

- A directed graph where edges are drawn from \(V_u\) to \(V_c\) and is represented as \(i \leftrightarrow j\);
- and \(|x_i - x_j|\) to denote the 2-norm distance between vertices \(i\) and \(j\);
- and \(d(i,j)\) to denote the graph distance between vertices \(i\) and \(j\).

User groups categorization criteria and energy levels

User Groups are to be categorized based on the set of Force laws that define a group. The Force laws we considered in HMM platform are: Optimal distance, Relative strength, Initial step size, Adaptive cooling, Area, Gravity, Scaling, Speed, Quadtree max level and Theta; for more information about each parameter, refer to (Fruchterman and Reingold 1991).

For evaluating, we defined the Spring systems (attractive systems) and Electrical forces (repulsive systems), refer to (Fruchterman and Reingold 1991) as well.

User groups method

It may be helpful to envision this model as a physical design. To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system. The vertices are placed in some initial layout and then released so that the spring forces on the rings move the system to a minimal energy state. Two practical adjustments are made to this idea: firstly, logarithmic strength springs are used; that is, the force exerted by a spring is:

\[
c_1 \times \log \left( \frac{d}{c_2} \right),
\]

where \(d\) is the length of the spring, and \(c_1\) and \(c_2\) are constants. Experience shows that Hooke’s Law springs are too strong when the vertices are far apart; the logarithmic force solves this problem. Note that the springs exert no force when \(d = c_2\). Secondly, we make nonadjacent vertices repel each other. An inverse square law force,

\[
c_3/d^2,
\]

where \(c_3\) is constant and \(d\) is the distance between the vertices, is suitable. The virtual equivalent of this mechanical system is simulated by the SPRING Algorithm, as shown below.

```
SPRING ALGORITHM (G: graph);
place vertices of G in random locations; {
repeat M times {
    calculate the force on each vertex;
    move the vertex \(c_4 \times \text{(force on vertex)}\);
}
draw graph on CRT or plotter
}
```

Algorithm 1. Spring Algorithm
The values $c_1 = 2; c_2 = 1; c_3 = 1; c_4 = 0.1$, are appropriate for most graphs. Almost all graphs achieve a minimal energy state after the simulation step ran 100 times ($M = 100$).

The repulsive force, $f_r$, exists between any two users $i$ and $j$, and is inversely proportional to the distance between them. The attractive force, $f_a$, on the other hand, exists only between any two users who have close similarities. We call it as neighboring users in this paper, and it is proportional to the square of the distance:

$$f_r(i, j) = -c \times k^2 / ||x_i - x_j||, i \neq j, i, j \in V,$$
$$f_a(i, j) = ||x_i - x_j||^2 / k, i \leftrightarrow j,$$

So, the combined force on a vertex $i$ is given as $f(i, x, k, c) = \sum_{i \leftrightarrow j} \frac{-c \times k^2}{||x_i - x_j||} (x_i - x_j) + \sum_{i \leftrightarrow j} \frac{||x_i - x_j||}{k} (x_i - x_j)$.

In the above formula, $k$ is a parameter known as the optimal distance; where parameter $C$ regulates the relative strength of the repulsive and attractive forces. So, if we have two vertices linked by an edge in the graph then if the distance between them is equal to the force between them diminishes, $k(c)^{1/3}$. The total energy of the system can be considered as,

$$Energy_{se}(x, k, c) = \sum_{i \in V} f^2 (i, x, k, c),$$

where $x$ is the vector of coordinates, $x = \{x_i | i \in V\}$.

Accordingly, we obtain the energy level for each community formed and then apply the Adaptive Cooling scheme, which takes the initial step size and step ratio as parameters to run the layout algorithm. Adaptive Cooling scheme (Hu 2005) is used to help the layout algorithm to avoid the energy local minima. Thus it ensures certain energy level is maintained for each user group and that energy value may increase or decrease but should lie within local max and local min values.

```plaintext
Adaptive Cooling ALGORITHM (G: graph);
ForceDirectedAlgorithm(G, x, tol); {
    converged = False;
    step = initial step length;
    Energy = Infinity;
    While (converged = False) {
        x0 = x; Energy0 = Energy; Energy = 0;
        for i \in V {
            f = 0; for (j \leftrightarrow i) f = f + \frac{f_a(i, j)}{||x_i - x_j||} (x_j - x_i);
            x_i = x_i + step * (f / ||f||^2);
        } step = update_step_length( step, Energy, Energy0);
        if (||x_i - x_j|| < K * tol) converged = TRUE;
    } return x;
}
```

Algorithm 2. Adaptive Cooling Algorithm

To update the step across the iterations we use update_step_length method based on the step_ratio given for the layout.
**Update_step_length ALGORITHM** (step, t, Energy, Energy<sub>0</sub>):

```java
if (Energy < Energy<sub>0</sub>) {
    progress = progress + 1;
    if (progress >= 5) {
        progress = 0;
        step = step/t;
    } else {
        progress = 0; step = t * step;
    }
} else {
    progress = 0; step = t * step;
}
```

**Algorithm 3. Update_step_length**

In the above function, progress is a static variable that is initialized to zero, and the parameter t is the step ratio that is one of the parameters given to the layout. The basic idea of the above algorithm is that the step length is kept unchanged if energy is being reduced, and is increased to step/t if the energy is reduced more than 5 times in a row. Thus, the step length is only reduced if the energy increases.

The other two forces related parameters we use in our system are the Quad max tree level and Theta. These are mainly the Barnes-Hut properties (Barnes and Hut 1986). These are used to determine the complexity of repulsive forces to $O(|V| \log |V|)$. An octree is taken as Quadtree in the context of 2D layout. An octree data structure is constructed by first forming a square that encloses all vertices. For these parameters the super node is defined in the cluster (Lewin 2013), which is assumed to situate at the center of gravity of the cluster, $x_s = (\sum_{i \in S} x_i)/|S|$. So, the repulsive force on vertex $i$ from this super node is

$$f_r(i, S) = -|S|C \times k^2 \frac{1}{||x_i - x_j||},$$

From (Quigley 2001; Tunkelang 1999) the super node $S$ is defined to be far away from vertex $i$, if the width of the square that contains the cluster is small, compared with the distance between the cluster and the vertex $i$,

$$\frac{d_S}{||x_i - x_S||} \leq \theta,$$

Above inequality is termed as Barnes-Hut opening criterion (Barnes and Hut 1986). The smaller the value of $\theta$, the more accurate the approximation to the repulsive force and more computationally expensive it is. To determine the maximum level to check the repulsive forces we use the parameter Quadtree max level and it is given by

$$h(max_{tree\,level}) = counts + \alpha \times n_s,$$

where counts is the total number of squares traversed, $n_s$ is the total number of super nodes found during one outer iteration, and $\alpha$ is a parameter that gives the best estimate of the CPU time.

Three parameters we use in Fruchterman Reingold (Fruchterman and Reingold 1991) force method are: Area (which defines number of nodes in the graph), Gravity (this attracts all nodes to the center to avoid dispersion of disconnected components) and Speed (convergence speed, whose increase may result in a precision loss).

**Additional Parameters used by ForceAtlas2 Algorithm**

ForceAtlas2 (Mathieu Jacomy 2011) needs few iterations to reach a significant quality. Each iteration is computed in a short amount of time. It is often 4 times more efficient than Yifan Hu. ForceAtlas2 benefits from the Barnes Hut optimization and the multithread implementation.
ForceAtlas 2 parameters
The first parameter is, LinLog Mode. It uses a logarithmic attraction force, as follows:
\[ f_a(n_1, n_2) = \log(1 + d(n_1, n_2)). \]
The second parameter is, Dissuade Hubs. Dissuade Hubs mode affects the shape of the graph by dividing the attraction force of each node by its degree plus one for nodes it points to. When active, the attraction force is computed as follows:
\[ f_a(n_1, n_2) = \frac{d(n_1, n_2)}{\text{deg}(n_1) + 1}. \]
This mode is meant to grant authorities (nodes with a high indegree) a more central position than hubs (nodes with a high outdegree). This is particularly interesting with social networks or web networks and in general in all directed networks where it is harder to be linked than to link. Dissuade Hubs tends to push hubs to the periphery while keeping authorities in the center of the graph.
The third parameter is, Prevent Overlap, with this mode enabled, the repulsion is modified so that the nodes do not overlap. The idea is to take in account the size of the nodes size \( n \) in computing the distance \( d(n_1, n_2) \) both in the attraction force and in the repulsion force.
The fourth parameter is, Approximate Repulsion. This is applied in the same manner as in Barnes-Hut optimization techniques (Barnes, 1986) to reduce complexity.

Calculating modularity
The quality of the clusters that are obtained from the partition of a network into communities of densely connected nodes is often measured using modularity parameter. The modularity of a partition is a scalar value between \(-1\) and \(1\) that measures the density of links inside the communities as compared to links between communities (Girvan, 2002; Newman, 2003; Newman, 2004). In the case of weighted networks (in this case the number of visits as the edge weight in our graph), modularity is defined as:
\[
Q = \frac{1}{2m} \sum_{n=0}^{n} \left[ A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),
\]
where \( A_{ij} \) represents the weight of the edge between \( i \) and \( j \); \( k_i = \sum_j A_{ij} \), is the sum of the weights of the edges attached to vertex \( i \) and \( c_i \) is the community to which vertex \( i \) is assigned, the \( \delta \)-function \( \delta(u, v) \) is 1 if \( u = v \) and 0 otherwise and \( m = \frac{1}{2} \sum_{ij} A_{ij} \).

Experimental Analysis
Based on the available data one can calculate the modularity for each force directed method and visualize the network across the three different algorithms: (i) Yi Fan Hu algorithm, (ii) Fruchterman and Reingold algorithm, and (iii) ForceAtlas2 algorithm. We used a subset of HyperManyMedia’s log (Logs, 2006) that consists of 8,501 Nodes and 12,827 edges.

Force Directed method (Fruchterman Reingold Algorithm)
Three parameters are used in the Fruchterman Reingold’s directed-force method: (i) Area (which defines the number of nodes in the graph), (ii) Gravity (which works to attract all nodes to the center to avoid dispersion of disconnected components); and (iii) Speed (convergence speed). Table 1 shows the results of 20 trials with the best results obtained in trial 6 having a modularity of 0.606 and number of communities (clusters) of 14. The metrics listed in Table 1 below are: Area (A), Gravity (G), Speed (Sp), Modularity (M), and the number of communities (C).

<table>
<thead>
<tr>
<th>A</th>
<th>G</th>
<th>Sp</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15000</td>
<td>10</td>
<td>50</td>
<td>0.597</td>
</tr>
<tr>
<td>2</td>
<td>12900</td>
<td>10</td>
<td>50</td>
<td>0.602</td>
</tr>
</tbody>
</table>
Table 1. Force-directed method (Fruchterman Reingold)

Figure 3 shows the partitions (clustered) obtained from Fruchterman Reingold force directed method and the percentages of nodes defined for each cluster.

Figure 3. Force Directed method Clusters

Figure 4 shows the best layout generated by Fruchterman Reingold (Fruchterman and Reingold 1991) force directed method with unique coloring for each cluster.
Table 2 below shows the results of 20 trials, with the best results obtained in trial 18 with a modularity of 0.610 and a number of communities (clusters) of 14. We noticed that we got the best results in this method when there is (i) a little repulsive force given by Scaling and (ii) a higher attractive force given by Gravity. The metrics listed in Table 2 are: Dissuade Hubs (D), LinLog Mode (L), Prevent Overlap (P), Edge Weight Influence (E), Scaling (S), Gravity (G), Tolerance (T), Approximate Repulsion (AR), Approximation (App), and the Number of Communities (C).

<table>
<thead>
<tr>
<th></th>
<th>D</th>
<th>L</th>
<th>P</th>
<th>E</th>
<th>S</th>
<th>G</th>
<th>T</th>
<th>AR</th>
<th>App</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>0.5</td>
<td>No</td>
<td>1.2</td>
<td>0.597</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>0.5</td>
<td>No</td>
<td>1.2</td>
<td>0.598</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>0.5</td>
<td>No</td>
<td>1.2</td>
<td>0.605</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>0.5</td>
<td>No</td>
<td>1.2</td>
<td>0.604</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>0.8</td>
<td>No</td>
<td>1.2</td>
<td>0.604</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>7</td>
<td>15</td>
<td>1</td>
<td>No</td>
<td>1.2</td>
<td>0.603</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>10</td>
<td>20</td>
<td>0.5</td>
<td>No</td>
<td>1.2</td>
<td>0.608</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>1</td>
<td>15</td>
<td>25</td>
<td>0.7</td>
<td>Yes</td>
<td>1.2</td>
<td>0.603</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>1</td>
<td>10</td>
<td>30</td>
<td>0.8</td>
<td>Yes</td>
<td>1.2</td>
<td>0.604</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
<td>15</td>
<td>30</td>
<td>0.5</td>
<td>Yes</td>
<td>1.2</td>
<td>0.597</td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
<td>20</td>
<td>30</td>
<td>0.5</td>
<td>Yes</td>
<td>1.2</td>
<td>0.605</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>1</td>
<td>20</td>
<td>20</td>
<td>0.7</td>
<td>Yes</td>
<td>1.2</td>
<td>0.601</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>20</td>
<td>15</td>
<td>0.7</td>
<td>Yes</td>
<td>1.2</td>
<td>0.603</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>1</td>
<td>15</td>
<td>20</td>
<td>0.5</td>
<td>No</td>
<td>0.9</td>
<td>0.597</td>
<td>14</td>
</tr>
<tr>
<td>15</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>1</td>
<td>30</td>
<td>15</td>
<td>0.5</td>
<td>No</td>
<td>0.9</td>
<td>0.603</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>1</td>
<td>10</td>
<td>25</td>
<td>0.2</td>
<td>No</td>
<td>0.5</td>
<td>0.601</td>
<td>14</td>
</tr>
</tbody>
</table>
Table 2. Force Directed method (Force Atlas 2)

The same process was repeated with the ForceAtlas 2 Algorithm (Mathieu Jacomy 2011) which is a continuous algorithm. It allows you to manipulate the graph while it is rendering (a classic force-vector, like Fruchterman Rheingold). This algorithm uses Barnes-Hut (Barnes 1986) optimization techniques and along with its own repulsive and tolerance levels. The best results in force-directed method are obtained when there is a little repulsive force given by Scaling and higher attractive force given by Gravity. Partitions obtained from Force Atlas Layout are shown in Figure 5 followed by percentages of nodes defined for each cluster.

![Figure 6. Force Atlas method Clusters](image)

Figure 6. Force Atlas method Clusters

Figure 6 shows the best layout generated by Force Atlas method with unique coloring for each cluster.

![Figure 6. Force Atlas method Layout](image)

Figure 6. Force Atlas method Layout
Force Directed method (Yi-fan Hu Algorithm)

Table 3 below shows the results of 20 trials, with the best results obtained in trial 12. This had a modularity of 0.607 and a number of communities (clusters) of 15. The metrics listed in Table 3 are: Quad Tree Max Level (QT), Theta (Th), Optimal Distance (OD), Relative Strength (RS), Initial Step Size (I), Step Ratio (SR), Adaptive Cooling (AC), Modularity (M) and the Number of Communities (C).

<table>
<thead>
<tr>
<th></th>
<th>QT</th>
<th>Th</th>
<th>OD</th>
<th>RS</th>
<th>I</th>
<th>SR</th>
<th>AC</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1.2</td>
<td>100</td>
<td>0.2</td>
<td>40</td>
<td>0.95</td>
<td>Yes</td>
<td>0.602</td>
<td>14</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>1</td>
<td>150</td>
<td>0.2</td>
<td>40</td>
<td>0.95</td>
<td>Yes</td>
<td>0.603</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>1.2</td>
<td>150</td>
<td>0.4</td>
<td>50</td>
<td>0.95</td>
<td>Yes</td>
<td>0.599</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>0.8</td>
<td>150</td>
<td>0.2</td>
<td>40</td>
<td>0.95</td>
<td>Yes</td>
<td>0.598</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>0.5</td>
<td>150</td>
<td>0.2</td>
<td>50</td>
<td>0.95</td>
<td>Yes</td>
<td>0.604</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>0.5</td>
<td>150</td>
<td>0.2</td>
<td>60</td>
<td>0.95</td>
<td>Yes</td>
<td>0.6</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>0.5</td>
<td>160</td>
<td>0.1</td>
<td>40</td>
<td>0.95</td>
<td>Yes</td>
<td>0.598</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>40</td>
<td>0.2</td>
<td>140</td>
<td>0.1</td>
<td>30</td>
<td>0.8</td>
<td>Yes</td>
<td>0.601</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>30</td>
<td>0.9</td>
<td>120</td>
<td>0.1</td>
<td>30</td>
<td>0.8</td>
<td>Yes</td>
<td>0.598</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>1</td>
<td>100</td>
<td>0.3</td>
<td>40</td>
<td>0.9</td>
<td>Yes</td>
<td>0.601</td>
<td>14</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
<td>1.2</td>
<td>120</td>
<td>0.5</td>
<td>30</td>
<td>0.9</td>
<td>Yes</td>
<td>0.597</td>
<td>15</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>1.2</td>
<td>120</td>
<td>0.2</td>
<td>40</td>
<td>0.95</td>
<td>Yes</td>
<td>0.607</td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>35</td>
<td>1.4</td>
<td>120</td>
<td>0.1</td>
<td>40</td>
<td>0.98</td>
<td>Yes</td>
<td>0.598</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>40</td>
<td>1.3</td>
<td>140</td>
<td>0.2</td>
<td>50</td>
<td>0.99</td>
<td>Yes</td>
<td>0.601</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>1.2</td>
<td>140</td>
<td>0.3</td>
<td>35</td>
<td>0.95</td>
<td>Yes</td>
<td>0.602</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>30</td>
<td>1.5</td>
<td>80</td>
<td>0.5</td>
<td>10</td>
<td>0.95</td>
<td>Yes</td>
<td>0.601</td>
<td>15</td>
</tr>
<tr>
<td>17</td>
<td>40</td>
<td>1</td>
<td>100</td>
<td>0.2</td>
<td>20</td>
<td>0.95</td>
<td>Yes</td>
<td>0.6</td>
<td>14</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>0.7</td>
<td>120</td>
<td>0.1</td>
<td>30</td>
<td>0.6</td>
<td>Yes</td>
<td>0.603</td>
<td>14</td>
</tr>
<tr>
<td>19</td>
<td>40</td>
<td>0.5</td>
<td>140</td>
<td>0.3</td>
<td>20</td>
<td>0.8</td>
<td>No</td>
<td>0.605</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>50</td>
<td>0.4</td>
<td>180</td>
<td>0.8</td>
<td>40</td>
<td>0.9</td>
<td>Yes</td>
<td>0.6</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 3. Force Directed method (Yi-fan Hu)

Finally, the Yi-Fan Hu Layout (Hu 2005) was run yielding a larger number of clusters compared to above two methods, but with loss in modularity. Partitions obtained from Yi-Fan Hu Layout are shown in Figure 7 followed by percentages of nodes defined for each cluster.
Figure 7. Yi Fan Hu method Clusters

Figure 8 shows the best layout generated by Yi Fan Hu method with unique coloring for each cluster.

Figure 8. Yi Fan Hu method Layout

Results
By running multiple trials, the best configuration of parameters for each algorithm was discovered. This is defined as the one that yielded the highest modularity structure. However, a slight difference in the results was noticed with regards to the number of clusters from each layout. The highest number of clusters was obtained by using the Yi’fan Hu algorithm, whereas a better modularity measure was obtained by using the Force Atlas 2 algorithm as shown in Table 4. In addition to these automatic methods of clustering, a manual process was conducted by looking inside each community separately to find the best coherent communities of resources. As a result, the communities generated by applying Force Atlas 2 were the most coherent among the three algorithms (i.e., one community consisted of Math, Chemistry, and Biology resources, another consisting of History and English resources, etc. The process was semi-automatic, meaning that we did not only rely on modularity measure, but also on our manual analysis of nodes inside each community.
### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Modularity</th>
<th># of Communities</th>
<th>μ</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fruchterman and Rheingold</td>
<td>0.606</td>
<td>14</td>
<td>0.601</td>
<td>0.003</td>
</tr>
<tr>
<td>Force Atlas 2</td>
<td><strong>0.610</strong></td>
<td><strong>14</strong></td>
<td><strong>0.603</strong></td>
<td><strong>0.004</strong></td>
</tr>
<tr>
<td>Yi-Fan Hu</td>
<td>0.607</td>
<td>15</td>
<td>0.601</td>
<td>0.003</td>
</tr>
</tbody>
</table>

*Table 4. Evaluation of the three Force-directed Methods (Best trials)*

### Designing a Social Recommender System

We propose adding a social recommender system the repository where recommendations are provided to a user (learner) based on detecting triangles in the community. Serrou et al (Serrou et al. 2010) defines the process of detecting triangles in a community as follows:

Let $G = (V,A)$ be a weighted undirected graph representing a complex network, where $V$ represents the vertices set and $A$ the edges set.

The objective is to identify communities of triangles, i.e. a partition with the requirement that the density of triangles formed by any three nodes $i,j$ and $k$ inside the same module is larger than the triangles formed outside the module. We used the Triangle modularity as defined by (Serrou et al. 2010) to build a social recommender system. The process is described as follows: (i) detecting the user's community; (ii) finding the closest Triangle; and finally, once the Triangle has been detected, the user is recommended resources from the three users that form the Triangle, as shown in SNA Recommender System Algorithm 4.

\[
SNA \text{ Recommender System} \quad \text{ALGORITHM} \quad (G,U,P, \text{Rec});
\]

\[
G = (V,A); \quad \text{(weighted directed graph)}
\]

\[
LR_s = \{LR_1, LR_2, ..., LR_n\}; \quad \text{(learning resources)}
\]

\[
P = \text{UserProfile}; \quad \text{(previously visited learning resources)}
\]

\[
\text{Rec} = LR_s; \quad \text{(recommended learning resources from the closest Triangle T)}
\]

Read network $G$ and UserProfile $P$

\[
\{
\]

Map $P$ to a community $g$ (subgraph = cluster) in $G$

Find the closest Triangle $T$ to user $P$ within $g$

\[
\text{Rec} = \{ LR \in \mu_i \mid \mu_i \in T\};
\]

while $\text{Rec} \neq \emptyset$ do \{ 

\[
\text{if} \ \text{Rec} \ni \in P \ \text{then} \ P = P \cup \text{Rec};
\]

\}

\}

*Algorithm 4: SNA Recommender System*
Figure 9 illustrates the implementation of SNA Recommender System Algorithm on HyperManyMedia platform, where we used the community detection method was used to provide recommendations of resources to a learner based on his/her profile in its community.

Figure 9. HMM Recommender System

Conclusion

The identification of community structure is one of the fundamental questions in the analysis of large scale complex networks. In this work, we propose a novel approach to extracting communities within a large network of cyberlearners and learning resources. The technique used is a heuristic which initially performs clustering using force-based visualization algorithms and then relies on network modularity to select good decompositions from those found visually. Through testing, we have determined appropriate parameters for optimal performance. At first we needed to choose the criteria on which we can categorize each community and then look at various options available for each community and then proceed accordingly. We defined the set of various force laws that are to be considered for the categorization of the communities. Once the communities have been recognized, we needed to make sure that each one has its own energy state, which determined the relevance level of that particular community in its range of proximity. Accordingly, we kept a threshold such that every community should satisfy that value in order for it to be recognized as a proper community in the network. We found that Yifan Hu Layout Algorithm is both efficient and of high quality; it overcomes the local minimums, with Barnes and Hut octree technique, which approximates short and long range force efficiently. We also used the adaptive cooling schemes and general repulsive force models to develop the set of forces to be applied on the data set for formation of the communities.

Finally, we used the community detection method to design a visual recommender system to recommend learning resources to cyberlearners within the same community. The algorithm is used to design a dynamic social recommender system. In the future, we plan to complete our evaluation of the visual recommender system, using objective metrics as well as user testing and compare it to our existing semantic personalized recommender and retrieval system.
REFERENCES


