A Distributed Framework for Social Network Analysis and Visualization

Arroyo, Daniel Ortiz; Larsen, Henrik Legind; Davidsen, Søren Atmakuri

A Distributed Framework for Social Network Analysis and Visualization

Daniel Ortiz-Arroyo
do@es.aau.dk

Henrik Legind Larsen
hll@es.aau.dk

Søren Atmakuri Davidsen
sda@es.aau.dk

Computational Intelligence and Security Lab
Department of Electronic Systems
Aalborg University, Esbjerg, Denmark

ABSTRACT

This paper introduces a distributed component for social network analysis (SNA) and visualization. The component separates light-weight from heavy-weight computation measures within a client-server solution.

The component includes a browser-based client that allows its integration in larger projects. Our current protptype has been specially designed to ease its integration within the VIRTUOSO project [4]. The prototype will allow security analysts to quickly visualize social networks and perform certain measures on them.

1. INTRODUCTION

In the practical application of social network analysis, an important issue is how to present the results produced to the end users. Visualization is one common method employed in social network analysis to address this problem. There are plenty of network visualization applications and libraries such as Netdraw, Socnetv, Jung, Prefuse and Gephi, among others. However, most of these visualization libraries are designed for desktop applications.

Browsers have become one of the most popular graphical user interfaces since they are available not only in personal computers and tablets but also in mobile phones. Users ccommonly use them for searching, browsing, entering data in web portals and reading news.

The goal of the prototype presented in this paper is to create a browser-based network visualization tool that additionally to visualization provides the basic infrastructure for the analysis of social networks ina distributed environment.

In our recent research work on social network analysis, we have focused on two main aspects:

1) algorithms to identify sets of key players [3];

2) predicting unobserved connections between players [1].

Further, from our participation in the VIRTUOSO project [4] we have set up two goals for the prototype:

1) support for visualization in a light-weight browser-based environment

2) implementation of useful tools for intelligence officers.

The remainder of this paper is structured as follows: Section 2 introduces important network measures in a security context, Section 3 discusses the architecture of our prototype and in Section 4 future directions and conclusions are offered.

2. NETWORK MEASURES IN SECURITY

The types of network calculations (measures) we have implemented in our prototype can be classified as follows:

1) measures which rank nodes;

2) measures which rank edges.

The classification is very general, but fits well for our purposes. In the following paragraphs we give a short introduction to networks and some of the common measures employed in SNA.

The most common model of a network is an directed weighted graph $G(V, E, w)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices, $E = \{e_1, e_2, ..., e_m\}$ a set of edges, each edge being a triplet of the two connecting nodes and their weight $e_i = (u, v, w)$. In our prototype we consider both directed and undirected graphs, where the undirected is a special case where $(v, u, w) \in E \Leftrightarrow (u, v, w) \in E$. Likewise, we consider also unweighted graphs, which is a special case where $\forall(u, v, w) \in E : w = 1$. We can consider the neighborhood of a node $N(v)$ to be the set of nodes adjacent to v, and the number of nodes in this set $D(v)$.

2.1 Node Ranking Measures

In ranking nodes we consider two concepts: 1) the centrality of the node - i.e. how important is it to the functioning of the network; 2) the hierarchial position in the network.

Node centrality ranking: Node centrality is well-established within the social network analysis community. It deals with the problem of identifying individual nodes which are important within the network context and more generally to identify groups of important nodes. An example is degree centrality, which measures number of adjacent nodes. A high degree centrality means the node is likely to receive information flowing in the network, hence in an intelligence context, such node could be a good source of information. Some classic node centrality measures are seen in Table 1.

Measure	Definition
Degree centrality	$C_D(v) = \frac{D(v)}{n-1}$
Betweenness centrality	$C_B(v) = \sum_{u \in V} \sum_{z \in V} g_{uz}(v)$
Closeness centrality	$C_C(v) = \sum_{u \in V} s(v,u)$
Eigenvector centrality	$x_i = \frac{1}{\lambda} \sum_{j=1}^{n} A_{ij} x_j$

Table 1: Commonly used centrality measures and their definition. $g_{uz}(v) = 1$ if $v \in p(u,z)$ determines if if a node belongs to a shortest path.

To rank nodes by centrality measures, we normalize the centrality results to the unit-interval and use the result as their rank. In the special case of finding sets of key players [3], we consider the rank of a node to be in $\{0,1\}$ as seen in Eq. 1.

$$r_{skp}(v) = \begin{cases} 1 & \text{if } v \in \{\text{key players}\} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Node hierarchial position ranking: This type of ranking has to do with the organization of the network. If we consider the organization-chart of a company, it is usally organized with layers of hierarchies encompassing director, managers, workers, etc. In intelligence, this hierarchy is rarely known, so the idea is to extract it from available information.

One approach could be to use node centrality measures to convert the graph to a directed graph which could then be transformed to a hierarchial structuring. Another approach could be to use a directed network to build a measure of agony, then the hierarchy is defined as the minimum-agony tree which covers the network [2].

The hierarchy-detecting methods produce a network where each node has a label equivalent of the level in the hierarchy. When ranking, we use an unit-interval normalized version of this level as the rank of the node.

2.2 Edge Ranking Measures

By ranking edges we want to indicate that some edges are more interesting than others. In our prototype, we consider two meanings of edge ranking: 1) shortest path ranking; 2) predicted edges ranking. In the following we describe both types of edge ranking.

Shortest path ranking: We say a graph has a path from a source node s to a target node t if there is a sequence of edges connecting the nodes, $(s, u_0), (u_0, u_1), ...(u_{n-1}, t)$ through a finite set of interconnected nodes. The minimum-length path between two nodes is called the shortest or geodesic path, which we denote $p(s,t)$ and it's length $s(s,t)$.

In shortest path ranking, we rank the shortest path by using Dijkstra's algorithm, meaning, when ranking shortest path of nodes (s,t), any edge has a rank in $\{0,1\}$, as seen in Eq. 2. More generally we can consider algorithms for k-shortest paths to get an edge rank in the unit interval.

$$r_{s,t}(e) = \begin{cases} 1 & \text{if } e \in p(s,t) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Predicted edges ranking: Predicting edges in a network can formally be described as: Given a graph $G(V,E)$, where E represents the observed edges, how likely is that an unobserved edge $(v,u) \notin E$ may exist between an arbitrary pair of nodes (v,u). The general approach is to use a scoring function $score(v,u)$ for the unobserved edge, and by normalizing the scores we get the rank of the unobserved edge.

Several methods exists for predicting edges, for example the graph structural measure provided by Adamic/Adar [1], as seen in Eq. 3 and Eq. 4, which considers the number of common neighbors of a node-pair. $N(v)$ is the set of nodes connected by a direct edge to v and $D(z)$ is the degree of z as earlier defined.

$$\text{score}_{AA}(v,u) = \sum_{z \in N(v) \cap N(u)} \frac{1}{\log D(z)} \quad (3)$$

$$r_{AA}(e:(v,u)) = \frac{\text{score}_{AA}(v,u)}{\max_{(v,u) \in E} \text{score}_{AA}(v,u)} \quad (4)$$

Other methods for predicting edges are also worth mentioning that depend on data which is not available in our definition of a graph. For example the temporal edge prediction in [1], which works on temporal edges where a timestamp $t(e)$ is available.

3. DESIGN AND ARCHITECTURE

The basic architecture of the complete application is described in Figure 1. Open sources are public information sources available on the Internet, which are gathered through other processes, normalized and stored in a knowledge base (KB). Our prototype loads social network structures[1] and allows visualization of the networks and to perform calculations as mentioned in Section 2.

An important aspect of our prototype is that it runs in a distributed fashion partially server-side and partially client-side (in a browser).

The browser is the end-user GUI, where networks can be loaded (selected from available network views), and the user can indicate which calculations can be performed on the network.

When a network-load is requested, the server-side will supply the network to the browser. When a network calculation

[1]In future versions the networks will be extracted automatically from the Knowledge Base

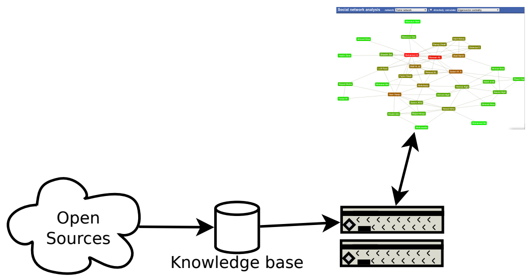

Figure 1: System architecture overview.

is requested, it can be completed only in the browser or in the server-side, depending on its complexity.

Figure 2 examplifies this flow:

1. Network data is extracted server-side and sent to browser for rendering.
2. End-user choose a server-side measure to evaluate on the network. The browser requests this to the server-side.
3. Server-side returns the results of the measure, browser updates the visualized network.
4. End-user choose a client-side measure to evaluate on the network. The browser calculates and updates the visualized network.

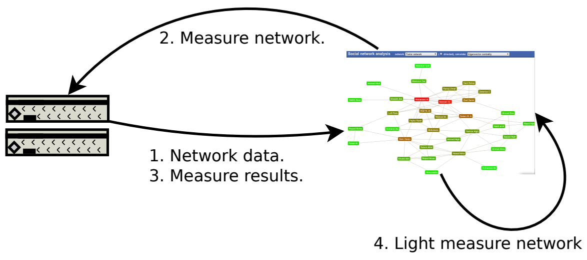

Figure 2: Prototype processing flow.

The current prototype performs shortest path and degree centrality measure calculations on browser-side, everything else is processed on the server-side.

The implementation of our prototype is done partially in Java language (server-side) and in Javascript (client-side).

In the *Server-side,* we use the popular Tomcat web-server together with Java servlets abd JGraphT and JGraphT-SNA libraries to represent and perform measures on the network.

On the *Client-side,* we use HTML5 as much as possible. Our graph-layout is based on Barnes-Hut[2] and web-workers to make the GUI reponsive to the end-user. The HTML5 canvas-element is used for rendering the network.

[2]Through the ArborJS library, `http://arborjs.org`

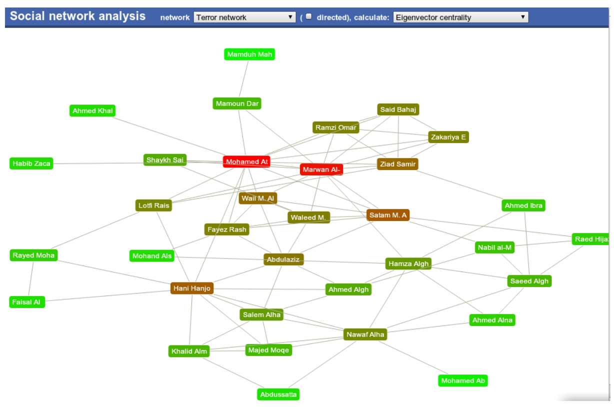
Social network analysis network Terror network (directed), calculate: Eigenvector centrality

Figure 3: Screenshot of prototype: Node ranking.

Figure 3 shows a sample screenshot of our prototype in action, where a node centrality measure has been calculated and more non-central to central nodes are displayed using a green to red gradient.

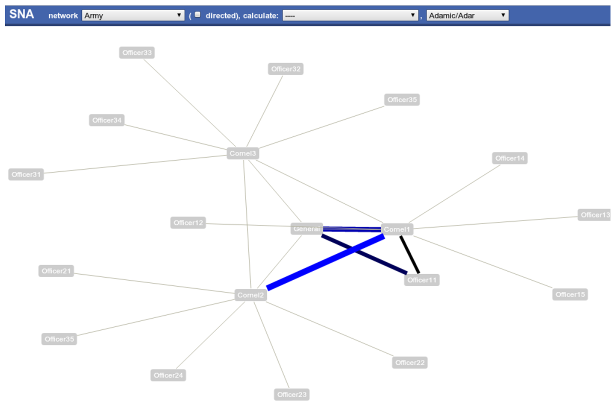
SNA network Army (directed), calculate: ---- , Adamic/Adar

Figure 4: Screenshot of prototype: Edge ranking.

Figure 4 is another sample screenshot from our prototype, in which an edge prediction measure has been calculated. The predicted edges are highlighted with stronger/wider edges for those which are considered more likely by the chosen algorithm.

4. FUTURE WORK AND CONCLUSIONS

While our prototype is already functional, it has still plenty of room for improvements and new features. In the VIRTUOSO project, the prototype will be integrated as a SOAP service that will emply the KB to extract networks in a JAVA-based portal environment.

In later versions of our prototype we use Java on the server side and the hybrid JAVA/Javascript framework Google Web Toolkit (GWT). GWT allows to write all code in JAVA language, and translate parts of the application to Javascript for execution on the browser. This is a good match for the current architecture of our prototype and will allow us to use a single language in development. This will have the additional advantage of making easier to move functionality from client to server and viceversa. For instance, measures which work directly on the network should all run in the browser and measures which require additional information, available only in the KB will run on the server.

Other features planned include allowing edition of the network by users, not to update the KB, but to ease the work of the intelligence officers that may want to perform *what if* questions and see how the measures change under conditions that are not available in the KB.

The current prototype offers only few options for working on really large networks. When extracting networks from the KB, the extraction will be made for a specific problem, which limits the size and order of the networks for visualization. One of the directions of our current research is to handle very large graphs using client-side techniques such as clustering nodes or adding zooming capabilities. On the server-side we would like to make use of a distributed infrastructure such as Hadoop's Map/Reduce parallelization, to make the user-interface work seamiglessly even with very complex and time-consuming network measures.

Our research is currently centered on hierarchy-detection methods.

To summarize, we have described a prototyope that employs a distributed architecture and includes features such as browser-based network visualization and SNA meatrics. The prototype helps intelligence officers to visualize and perform calculations on social networks. Finally we have discussed further development of the prototype.

5. REFERENCES

[1] S. A. Davidsen and D. Ortiz-Arroyo. *Centrality robustness and link prediction in complex social networks*, chapter 98. Computer Communications and Networks. Springer, 2011 (to appear).

[2] M. Gupte, P. Shankar, J. Li, S. Muthukrishnan, and L. Iftode. Finding hierarchy in directed online social networks. In *Proc. of the 20th international conference on World wide web*, WWW '11, pages 557–566. ACM, 2011.

[3] D. Ortiz-Arroyo and D. M. A. Hussain. An information theory approach to identify sets of key players. In *Proceedings of the 1st European Conference on Intelligence and Security Informatics*, EuroISI '08, pages 15–26. Springer, 2008.

[4] VIRTUOSO. *Versatile InfoRmation Toolkit for end-Users oriented Open-Sources explOitation.* EU Frame Program 7, 2010.