An Extensible Open-Source Framework for Social Network Analysis

Michal Barla and Mária Bieliková

Abstract Online communities that form social networks became extremely important in many tasks related with information processing, presentation, navigation, especially in context of web-based information systems. Web-based information systems employing communities could benefit from the classical studies of human social interactions – social network analysis. In this paper, we present an extensible open-source JAVA-based framework for social network analysis which can be used either a a stand-alone application with its own GUI as well as a library within third-party software projects developed in JAVA. We provide not only a standalone desktop application, but the whole framework, which allows anyone to incorporate results of social networks analysis into their own project, which would possibly boost up its functionality, enhance the results etc.

1 Introduction

Online communities became very trendy. Apart from its impact on our every day lives, with all blogs, wikis, tagged content, social portals and other Web 2.0 features, it is very popular also in the research community. As an example, we can take the recent research in personalization and adaptivity of web applications, with many papers leveraging the fact that an online user belongs to certain community.

Example of a system which employs community-based personalization is *Community-based Conference Navigator* [7]. It uses social navigation support to help conference attendees to plan their attendance at the most appropriate sessions and make sure that the most important papers are not overlooked. The system provides additional adaptive layer over the conference planning system, which allows

Institute of Informatics and Software Engineering,
Faculty of Informatics and Information Technologies, SUT in Bratislava
Bratislava, Slovakia,
e-mail: {name.surname}@fiit.stuba.sk

the conference attendees to browse the conference program and plan their schedule. Community-based conference navigator tracks different activities of the community (such as paper scheduling) and allows users to add comments to papers. All activities result in updates of the community profile, which accumulate over time the "wisdom" of the community, used in the adaptation of the original system. The selection of the community is done manually by each user. If the user does not find the suitable community, she is allowed to create a new one. Moreover, the user can switch between communities anytime during the usage of the system, which gives instantly the annotations for a different community.

However, it seems that the user can act only as a member of one community at a time, so all actions contribute only to one community profile. In reality, many people act as "bridges" between different communities, so it would not be easy for them to choose strictly only one of them. In other words, web-based information systems employing communities could benefit from the results of classical studies of human social interactions – social network analysis. Its goal is determining the functional roles of individuals (e.g., leaders, followers, "popular people", early adopters) in a social network and diagnosing network-wide conditions or states. So, in the case of community-based conference navigator, a paper scheduled by somebody who is considered as an authority should be considered as more important.

We realized that much of the research, similarly to the [7], stays on the group level and ignores information encoded in individual, one-to-one relationships, which can be extracted by applying social network analysis. One possible reason for such a situation can be a lack of proper software tools and development kits, which would facilitate the employment of social network analysis to the developers/researchers.

This is our motivation for designing and developing *Mitandao*, an open source software for social network analysis which can be used either as a stand alone application or as a library providing analytical services to other applications via API. These could be either classical metrics of social network analysis such as metrics for determining centrality (node degree, betweenness and closeness centrality etc.) or more recent methods of link analysis [4] such as PageRank. Moreover, we designed *Mitandao* as an extensible framework, where everyone can add his or her own modules for network analysis.

2 Related Work

As we mentioned already, social networks are gaining attention in various fields of research. Social navigation on the web [6] is an efficient method how to guide users to the content they might be interested in. Social networks are also used to overcome the initialization stage of recommender systems [8] or to solve the *cold-start* problem in user modeling [1, 11]. Other problem, which could benefit from the information stored within social networks is a disambiguation of person names, e.g., to normalize a publications database [10] or for the purpose of automatized infor-

mation extraction from the web [3]. All the aforementioned research areas could benefit from the reusable, interoperable and extensible social network analyzer.

Weka [14] is an open-source collection of machine learning and data mining algorithms written in JAVA. It is possible to use it as a standalone application or integrate some of its parts into another software project. It provides a basic support for distributed computing, which allows user to execute different setups of self-standing algorithms simultaneously. Since Weka is an open source software, it is possible to add a plugin (a new algorithm) into it or change an existing one according to user's needs. However, Weka is primarily targeted at data mining and its usage to social network analysis is not straightforward.

Ucinet [5] is a commercial software for social network analysis being developed by Analytic Technologies. It comes with a variety of functions to analyze the network (centrality measures, subgroup identification, role analysis etc.) and multiple ways of its visualization. The main drawback is that the software can be used only "as is". You need to have your data ready, import them to Ucinet, execute the chosen algorithm and observe the results. There is no way of modifying existing algorithms or adding your own one to the process. Moreover, Ucinet does not provide any kind of API which would allow other software to use its services.

InFlow (`orgnet.com/inflow3.html`) is another commercial software used to analyze social and organizational networks. Similarly to Ucinet, it contains a set of algorithms to analyze the structure and dynamics of the given network. What distinguishes InFlow is the "What-If" feature: every time the network changes, InFlow re-runs automatically selected analysis and display the results. This allows for efficient simulations, especially useful when designing an organizational structure within a company.

Pajek [2] is a program for analysis and visualization of large networks developed at University of Ljubljana. It is free for non-commercial use. It provides many algorithms and has a powerful drawing and layout algorithms. Unfortunately, similarly to Ucinet and InFlow, there is no way of calling the Pajek's logic from another program.

VisuaLyzer (`mdlogix.com/solutions/`) from Mdlogix is another commercial tool for analyzing and visualizing graphs. It has a clean and usable user interface which provides access to an interesting set of algorithms and visualizing options. Moreover, it allows the user to create and use new types of attributes for nodes and edges and is capable to use these attributes in algorithms, query functions for selecting nodes etc. Similarly to previously analyzed software tools, it does not provide any API, which would allow to use the features outside of the graphical user interface.

Overall, we can see that the majority of existing software solutions does not cover our requirements for an extensible and interoperable social network analyzer, such as possibility to be used automatically, by means of APIs. Second, their architecture and closed-source licensing does not allow ordinary users to add new functionality or change the existing one. The only exception is Weka, which is on the other hand too complex for the task of social network analysis.

3 Mitandao: Functionality and Architecture

All the functionality of *Mitandao* framework is concentrated in the execution of *workflows*. Each workflow consists of following stages:

1. import/use current graph
2. filtering
3. analysis
4. export/visualization

Workflows can be chained together, which allows for efficient application of any possible filter/analyze combination (e.g., we can compute betweenness centrality for every node in the first workflow and apply a filter of nodes based on this value in the second workflow, taking the graph from the first workflow as an input). Along with graph editing functions and undo/redo functions based on on-demand generated checkpoints, *Mitandao* provides all necessary methods for a successful dynamic analysis of any kind of social network.

Mitandao was designed with simplicity and extendability in mind. We separated completely the functionality of *Mitandao* from its GUI in order to allow anyone to use it as a library in his or her own project. Moreover, the *Mitandao* library provides only the core functionality required to execute a *workflow* along with some supporting tools, while the actual logic is separated into pluggable *modules*.

We recognize four types of modules according to four stages of a workflow:

1. *Input module* – used to load a graph (social network) into a working environment;
2. *Filter module* – used to remove nodes from the loaded graph according to the specific conditions;
3. *Algorithm module* – performing actual analysis (i.e., computation of metrics) of loaded social network;
4. *Output module* – used to store current graph in an external data-store.

Every module takes a graph as an input and returns another graph which is passed to the subsequent module or displayed on the screen. The input graph can be empty, which is often the case for the *Input modules*, which actually "creates" the graph for further processing. Not every module chained in a workflow needs to perform changes in the graph – the output graph can be equal to the input graph, i.e., when an *Output modules* does not change the graph itself, but dumps it to the file or database as a side effect.

For the internal graph representation we chose to use a widely used JUNG framework (Java Universal Network/Graph Framework, `jung.sourceforge.net`) for graph modeling and analysis. Use of the same graph structure ensures a high level of interoperability and usability of our framework. Moreover, we could take advantage of JUNG visualization framework and painlessly integrate JUNG implementations of various analysis algorithms.

JUNG is able to store custom data (e.g., analysis results) in the form of *key – value* pairs. To standardize and unify the way of storing various data in the graph, we took the idea of JUNG labeler (a structure, which applies a set of labels to a graph

to which it is attached to) and derived two labelers (for vertices and edges) which are to be used by all modules to store and manipulate custom graph data.

The library consists of following components (see Fig. 1):

- *Core component* – apart from definitions of basic components (such as workflow) or exceptions, which can occur during the analysis, it contains the entry interface for the whole library. The core contains a *ParameterReader* module, responsible for setting up individual modules according to the parameters chosen within the GUI and *ClassLoader* module, which looks-up available modules in pre-defined classpaths.
- *Graph component* – provides the basic utilities to manipulate graphs such as combining two graphs into one (union), copying data from one graph to another, storing custom data in the graph and additional converters.
- *Modules component* – contains interfaces of workflow modules, a *ModuleManager* for accessing all loaded modules. It contains an implementation of two basic input and output modules working with checkpoints (complete dump/recovery of a graph to/from a file).
- *UI Framework* – Every module author has an option to annotate (using JAVA Annotations) the module parameters which require user input (e.g., an input module might require a name of the input file). *UI Framework* uses such annotations to generate the GUI for setting these values and ensures proper setting of module variables via setter methods. Alternatively, the module author can provide his own GUI for setting the module parameters. In such case, *UI Framework* just passes a map of user supplied values to the module.

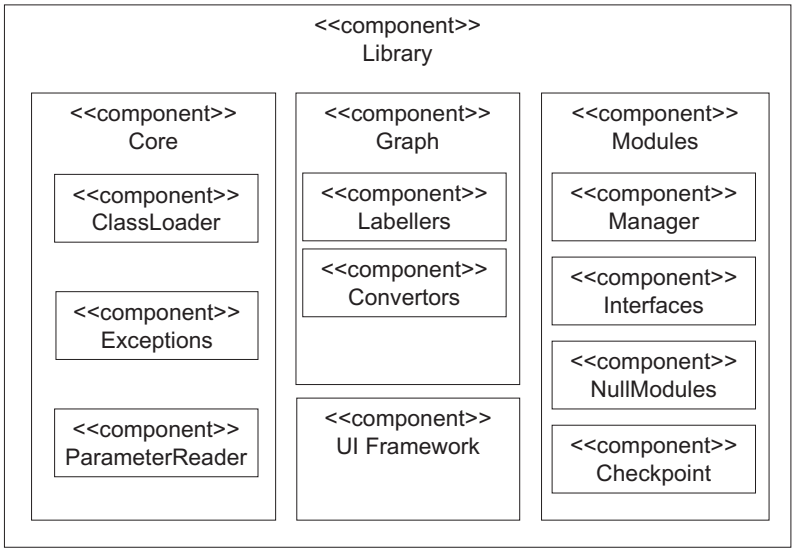

Fig. 1 Architecture of the *Mitandao* library

4 Mitandao: Modules

All the functionality provided by *Mitandao* is provided through modules combined into workflows. We decided to support various types of inputs such as Pajek and GraphML input files and provide also means to import data from a relational database, with either flat table structure or a structure using a join table (e.g., for bipartite graph where relationships are modeled via another entity). After analysis, when needed, the social network (graph) can be saved (exported) into GraphML or Pajek files.

After loading a social network into working environment, *Mitandao* provides modules for its optional filtering. Like that we can remove isolants (nodes with no connections at all) or use a more complex filter, which allows user to setup multiple conditions on a particular attribute of nodes.

The analysis modules are responsible for adding new properties to the social network, its nodes and connections between them. Currently implemented analysis modules are mainly wrappers on the top of JUNG library, providing the user with easy way how to compute various attributes of the nodes such as betweenness centrality or degree distribution.

Every stage of the workflow (input,filter, analysis, output) is optional and workflow can be executed without it. So, if an input module is not used to read a graph from the external storage, the subsequent stage receives an empty graph to work with. Similarly, user can choose not to use any output module and to process the resulting graph by other means.

5 Mitandao as a Standalone Application

Mitandao application is a java-based application built on the top of the *Mitandao* library. It provides two basic ways of setting up a workflow:

- *classical tabbed wizard* with pre-defined simple linear analysis consisting of one input, one optional filtering algorithm, one optional analytical algorithm and one output (with optional graph visualization at the end). Each step is presented in one tab containing all controls necessary to choose the appropriate module and to setup its parameters;
- *graphical wizard*, which allows to create advanced analysis with multiple input, filtering, analytical and output stages. Moreover, it allows to create forks and joins in the workflow, thus performing multiple branches of analysis or some of its stages.

When a graph is displayed (Fig. 2), a user has possibilities to explore it freely by zooming in and out and moving the graph. The user can drag nodes to another position, explore the properties of nodes and edges, add/delete nodes/edges. It is also possible to omit the input module stage and start the analysis with an empty graph, fill it interactively by nodes and edges and assign various attributes to them.

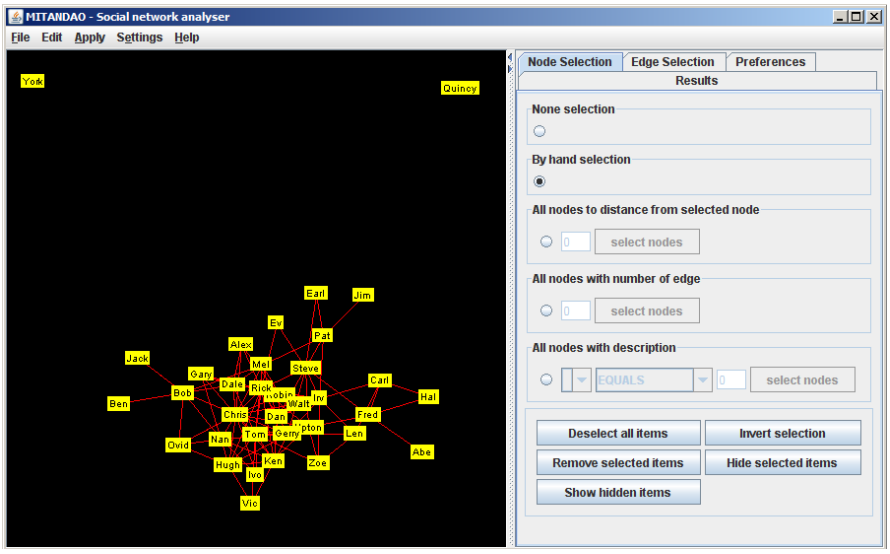

Fig. 2 Screenshot of the main *Mitandao* window. The left part shows the current graph, while the panel on the right contains controls to select nodes or edges and to examine the attributes of the selected items.

Then, the user can store this manually created graph for the future reference or use it as an input for a new workflow.

Node and edge selection of the displayed graph can be done by either manually clicking the nodes to be selected or by using pre-defined selectors, which allows for selecting all nodes within a defined distance from an already selected node, selecting nodes with a certain degree or selecting nodes/edges according to the values of their attributes.

6 Conclusions

In this paper, we described *Mitandao*, an extensible open source social network analysis framework and an application built on top of it. The most important part is that we provide not only a standalone desktop application, but the whole framework, which allows anyone to incorporate results of social networks analysis into their own project, which would possibly boost up its functionality, enhance the results etc.

We already started to incorporate results of social network analysis coming from *Mitandao* framework into our research. It is suitable mostly for tools realizing web search and navigation [9, 12, 13]. In [1], we proposed a method for initializing user model of a new user by leveraging his or her social connections to other users. Our approach requires that every link between two users is assigned a weight and every

user (node in the network) is assigned a rank. All these values come as results of automated invocation of analysis workflows within our *Mitandao* framework.

The framework (source codes, javadoc documentation and usage guides) is available at `mitandao.sourceforge.net`.

Acknowledgment

This work was partially supported by the Cultural and Educational Grant Agency of the Slovak Republic, grant No. KEGA 3/5187/07 and by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0508/09.

We wish to thank students Lucia Jastrzembská, Tomáš Jelínek, Katka Kostková, Luboš Omelina and Tomáš Konečný for their invaluable contribution to the project.

References

1. Barla, M.: Leveraging Social Networks for Cold Start User Modeling Problem Solving. In: M. Bieliková (ed.) IIT.SRC 2008: Student Research Conference, pp. 150–157 (2008)
2. Batagelj, V., Mrvar, A.: Pajek - Analysis and Visualization of Large Networks. In: Graph Drawing, LNCS 2265, pp. 8–11. Springer (2002)
3. Bekkerman, R., McCallum, A.: Disambiguating Web appearances of people in a social network. In: A. Ellis, T. Hagino (eds.) WWW 2005, pp. 463–470. ACM (2005)
4. Bing, L.: Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data. Springer Berlin Heidelberg (2007)
5. Borgatti, S., Everett, M., Freeman, L.: UCINET 6.0. Analytic Technologies (2008)
6. Brusilovsky, P.: Social information access: The other side of the social web. In: V. Geffert, et al. (eds.) SOFSEM 2008: Theory and Practice of Computer Science, LNCS 4910, pp. 5–22. Springer (2008)
7. Farzan, R., Brusilovsky, P.: Community-based Conference Navigator. In: J. Vassileva, et al. (eds.) Socium: Adaptation and Personalisation in Social Systems: Groups, Teams, Communities. Workshop held at UM 2007, pp. 30–39 (2007)
8. Massa, P., Bhattacharjee, B.: Using Trust in Recommender Systems: an Experimental Analysis. In: 2nd Int. Conference on Trust Management (2004)
9. Návrat, P., Taraba, T., Bou Ezzeddine, A., Chudá, D.: Context Search Enhanced by Readability Index. In: IFIP 20th World Computer Congress, TC 12: IFIP AI 2008, pp. 373–382. Springer Science+Business Media, IFI Series, Vol. 276, Milano, Italy (2008)
10. Reuther, P., et al.: Managing the Quality of Person Names in DBLP. In: J. Gonzalo, et al. (eds.) ECDL 2006, LNCS 4172, pp. 508–511. Springer (2006)
11. Tsiriga, V., Virvou, M.: A Framework for the Initialization of Student Models in Web-based Intelligent Tutoring Systems. User Model. User-Adapt. Interact. **14**(4), 289–316 (2004)
12. Tvarožek, M.: Personalized Navigation in the Semantic Web. In: V. Wade, H. Ashman, B. Smyth (eds.) 4th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems, AH'06, pp. 467–471. Springer, LNCS 4018, Dublin, Ireland (2006)
13. Tvarožek, M., Barla, M., Frivolt, G., Tomša, M., Bieliková, M.: Improving Search in the Semantic Web via Integrated Personalized Faceted and Visual Navigation. In: V. Geffert, et al. (eds.) SOFSEM 2008, LNCS 4910, pp. 778–789. Springer (2008)
14. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, 2nd Edition. Morgan Kaufmann (2005)