networks.tb

A free software suite for co-evolving network analysis

Reference document v0.21

(partial documentation)

Camille Roth*

April 12, 2006

Copyright/copyleft

1 networks.tb: a toolbox for co-evolving networks

`networks.tb` is a software suite for analyzing co-evolving networks. It is recommended for the study of socio-semantic networks, or epistemic networks. Actually, it has initially been designed for that purpose as an helper and empirical study tool for a knowledge community case study [3]. In particular, it is possible to work on epistemic networks consisting of scientists using concepts, with empirical data extrated from `MedLine` bibliographical data, through `networks.tb/medline.scanner`. More broadly, the suite is also relevant for Galois lattice (or concept lattice) computation and manipulation, using `networks.tb/galois` — see more specifically [4].

All `networks.tb` programs use the same file format, the `networks.tb` format, which is comprehensively detailed in Sec. 3. Thus, it is possible to work on the same empirical data with every `networks.tb` application, as well as manipulate data using proprietary software based on the `networks.tb` format. Basically, the `networks.tb` format consists of an index file, along with several other specific data files (agent list, concept list, matrices, etc.) listed in the index file; as such, many `networks.tb` applications need to be fed only by an index file.

`networks.tb` actually consists of the following different programs and libraries:

*CREA (Center for Research in Applied Epistemology), CNRS/Ecole Polytechnique, 1 rue Descartes, 75005 Paris, France. *camille.roth@polytechnique.edu* or *camille.roth@ponts.org*

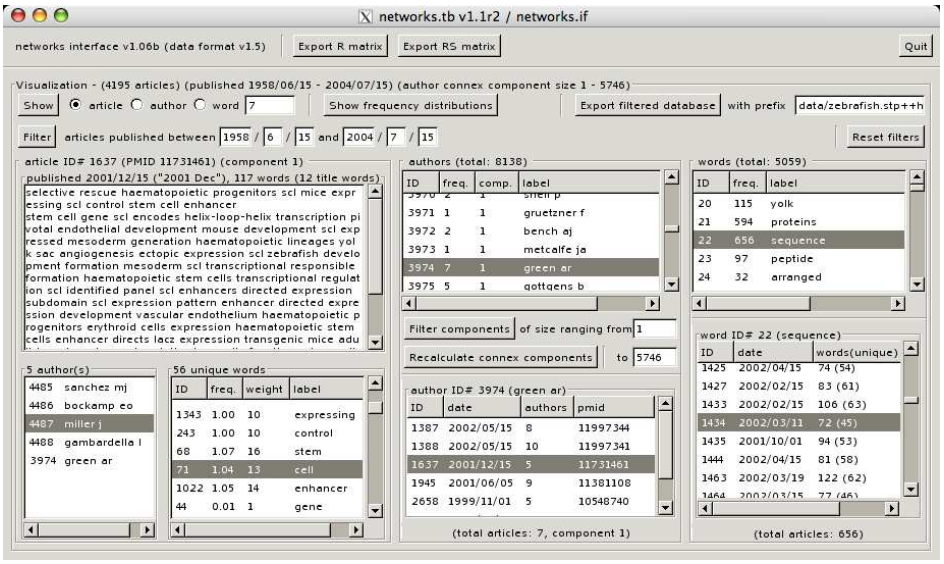

Figure 1: `networks.tb/networks.if`

1.1 `networks.tb/networks.if`

[as of v1.06b]

`networks.tb` is a graphical interface and front-end for epistemic network observation and manipulation. It should be launched using only one argument specifying a `networks.tb` index file. A screenshot is displayed on Fig. 1.
Source files: `networks.if.h`, `networks.if.c`, `networks.if.helper.h`, `networks.if.helper.c`

1.2 `networks.tb/galois`

[as of v1.24]

`networks.tb/galois` is an extensive Galois Lattice (GL, a.k.a. "concept lattice" [6]) computing application [4], including various low-level manipulations on relationship matrices that are used for creating Galois lattices.[1] This program is controlled by a text-based interface: `networks.tb/galois` is launched with one argument, the `networks.tb` index file corresponding to the binary relation matrix R. Then, the program successively asks several questions that guide the lattice computation:

1. Process the index file given as an argument, find the matrix file (which must have already been created, by `networks.tb/networks.if` for instance), then display various information on the corresponding bipartite graph between objects and properties (agents and concepts, rows and columns).

2. Propose methods for randomizing links in R: (1) keep only the same density of links, (2) keep the distribution of links from objects to properties (agents to concepts), (3) keep the same distribution of links [2].

[1] Note that all "lindig/*" files are part of `Concepts`, a library written by Christian Lindig. These files are free software, distributed under the GNU Public License version 2, and copyright (c) 1994 Technical University of Braunschweig, Germany.

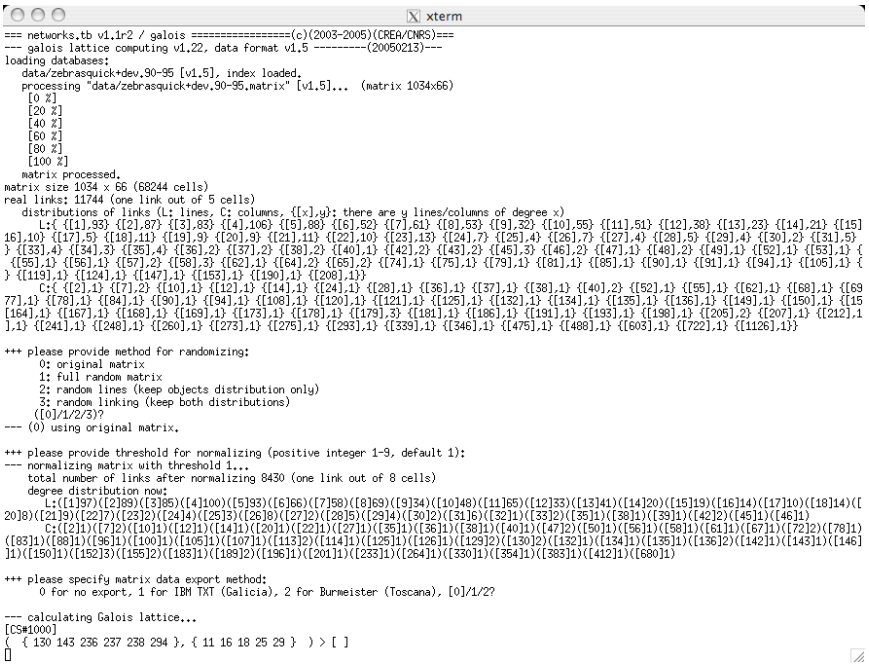

Figure 2: `networks.tb/galois`

3. Ask for a filtering threshold α, so that any weighted link in R becomes a binary (0/1) link if its weight is above α. Then, it is possible to export the binary relation matrix in various formats (either "`IBM TXT`" for use with `Galicia`, or "`Burmeister`" for use with `Toscana`).
4. Compute the GL.
5. Display the distribution of object set (agent set, item set, etc.) sizes of GL nodes (a.k.a. complete couples or closed couples), and exports it under "`galois.export.distrib`",
6. *Upon request*, compute various quantities on GL nodes (distances, simplified hierarchy) and exports this extended lattice under "`ccexport.galois`".

A screenshot of `networks.tb/galois` is shown on Fig. 2.

Source files: `galois.h, galois.c, lindig/{concept.h, concept.c, config.h, defines.h, list.c, list.h, panic.c, panic.h, relation.c, relation.h, set.c, set.h}`

1.3 `networks.tb/medline.scanner`

`[as of v1.19]`

A scanner and converter for `MedLine` databases, available for instance from *PubMed*. Creates `networks.tb`-compatible databases, directly viewable with `networks.tb/networks.if`. As shown on Fig. 3.
Source files: `medline.scanner.h, medline.scanner.c`
Needed files: `words.base(.dummy)`[2], `stop-words.base(.dummy)`[3]

[2]List of word classes relevant for the database – the format of this file is like one produced by `stembase`, [INDEX].words.base.

[3]Stop-word list: the words in this file will simply be ignored by `networks.tb/medline.scanner`.

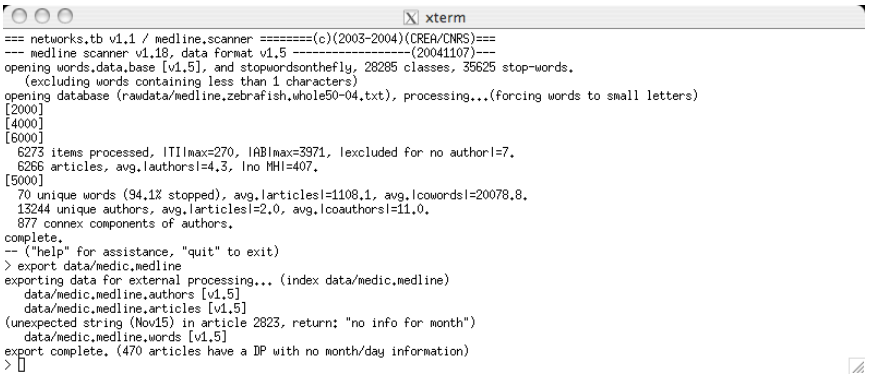

Figure 3: `networks.tb/medline.scanner`

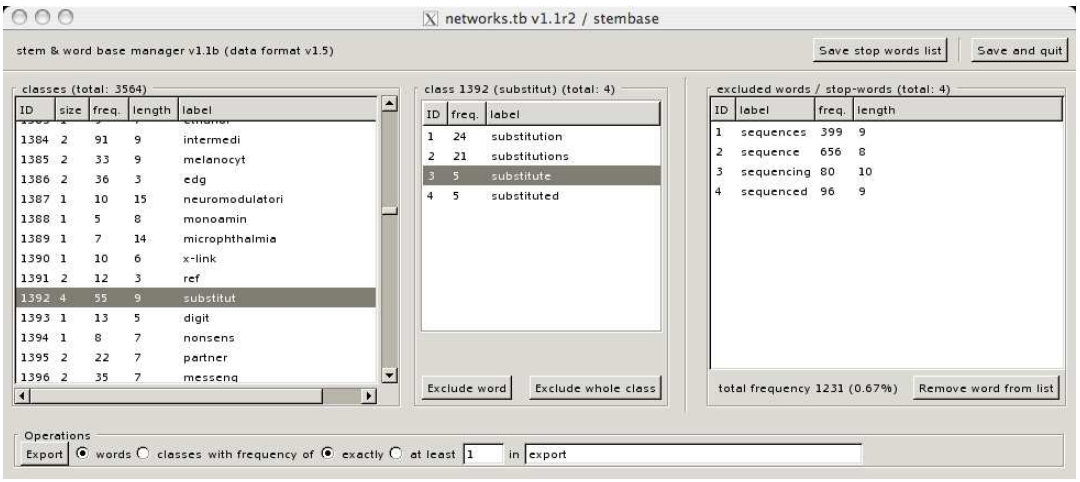

Figure 4: `networks.tb/stembase`

1.4 `networks.tb/stembase`

A vocabulary manipulation tool, strongly associated with `networks.tb/networks.if`. As shown on Fig. 4.
Source file: `stembase.c`

1.5 `networks.tb/networks.tb`

Functions and programs shared by `networks.tb` applications.
Source files: `networks.tb.h`, `networks.tb.c`, `scanner.h`, `scanner.c`, `gtk.helper.h`, `gtk.helper.c`, `converter.c`, `networks.tb.filelist`, `networks.tb.fileformat`

2 Examples of procedures

2.1 Using database-relevant stop words and word classes with `networks.tb/medline.scanner`

Case: One has a `MedLine` database, in plain text, and wants to create a `networks.tb` database from this bibliographical data, as well as do a basic linguistic processing on the words present in the database: exclude stop words relevantly with respect to the field, and regroup words within proper word classes. The procedure is as follows:

1. Check that files `words.base` and `stop-words.base` are either empty, or pointing to the right database
2. Launch `medline.scanner` on the raw database, if successful *export* to `[INDEX]`
3. Lemmatize `[INDEX].words` to `[INDEX].words.stemmed`, preferably through `porter/porter.c`
4. Launch `stembase` to create word classes into `[INDEX].words.base`, export n-hapaxes (with n <threshold), and exclude non-relevant word classes (deemed non-relevant by an expert)
5. Create an exhaustive stop-words list by concatenating `stopwords++.txt`[4] with n-hapaxes and non-relevant word classes[5], then link `stop-words.base` to this file
6. Regroup or split some classes directly in `[INDEX].words.base`, then link `words.data.base` to this file
7. Once again, launch `medline.scanner`, *export* again into the final processed database.

3 File format

`[this section applies for file format v1.5, used by networks.tb v1.1]`

The file format commonly used by `networks.tb` applications is actually a set of file formats, driven by a general index file, which is also the basename for all files related to this index file. For instance, for an index file named `[INDEX]`, you may find the list of authors under `[INDEX].authors`, the list of words under `[INDEX].words`, etc. The first line for every file corresponds to the version of the file format, it is thus easy to check whether a given database is compliant or not with the version of `networks.tb` being used.

3.1 General index file

`[INDEX]:` the index file contains the list of multiple files for a given dataset. The first line gives the format version. Then, a series of pairs of lines describes each file contained in the dataset: the first line of a couple is a number, specifying the type of the file, and the second line is the path and filename of that file. Number 0 indicates an author file (usually `[INDEX].authors`); 1, the list of articles (or events, usually `[INDEX].articles`); 2, the list of words (usually `[INDEX].words`); 4, the matrix authors/words, to be used with `networks.tb/galois` (usually `[INDEX].matrix`); 5 and 6 refer to the database of words and stop-words respectively, to be used with `networks.tb/stembase` (usually `[INDEX].base` and `[INDEX].stop`). For example:

[4] This file contains standard rhetorical and general scientific English. It is made of `stopwords.txt` (a basic stop-word file containing 337 basic stop-words such as *of*, *the*, etc.) and `stop.words.2620` (an additional file containing 2620 usual language stop-words, such as *new*, *number*, etc. and general scientific vocabulary, such as *study*, *article*, etc.).

[5] In general, a list of stop-words that is appropriate for the database. For instance, `stopwords++zeb.txt` contains a list of words that are not significant for zebrafish-related databases.

```
1.5
0
data/zebrafish.authors
1
data/zebrafish.articles
2
data/zebrafish.words
```

3.2 Item files

`[INDEX].authors:` contains author names. The first line gives the file format version. On the second is the total number of authors. Finally, the list of authors is given, one per line (author #1, author #2, etc.). For example:

```
1.5
1140
schmidt_ma
ogez_jr
hodgdon_jc
```

Here, author #1 is "MA Schmidt", author #2 is "JR Ogez", etc.

`[INDEX].words:` contains the list of words. The first line gives the file format version. On the second is the total number of words. Finally, the list of words is given, one per line (word #1, word #2, etc.). For example:

```
1.5
6669
development
application
synthetic
```

Here, word #1 is "development", word #2 is "application".

3.3 Databases for events

The file `[INDEX].articles` contains the list of articles (or events) and mentions authors and concepts which co-appear in a given article, chronologically. The first line is the file version. On the second line comes the number of articles. Then, the following structure is repeated *for each article*:

1. PMID (PubMed ID).
2. UI (unique identifier).
3. publication date (string, that is: the string appearing in the original database).
4. publication date (integer, that is: a processed date from the string appearing in the original database — this could be 890701 for "1989 Jul 1", or 1010315 for "2001 Mar").[6]
5. number of words.
6. number of words in the title,

[6]Note that when the original date consists of a year only, the date is taken to by 1st of July. Similarly, when the original date is simply a month and a year, the day is taken to be 15.

7. followed by word IDs for each word (including title words, which come first).

8. number of authors,

9. followed by author IDs for each author.

3.4 Files proper to `networks.tb/galois`

Format of `galois.export.distrib`: The file consists of a single line, formatted as a set ("$\{d_1, d_2, ..., d_n\}$") where the i-th element d_i contains the number of GL nodes that gather $i-1$ objects. For example:

```
{1,21,50,62,47,39,21,22,10,11,4,5,4,5,3,0,0,1,1,1,1,1,0,0,1,0,0,2,3,1,0,0\
1,0,0,0,0,1}
```

Format of `ccexport.galois`: As is the case with most `networks.tb` files, the first line indicates the file format version. The following description is accurate as of file format v1.5.

The three first lines include the version number, the number of nodes in the lattice, and a correspondance marker.[7] Then, each node is being described by a set of lines, organized as follows:

1. concept number,

2. distance from the top,

3. value,[8]

4. contingency,[9]

5. number of sons,

6. size of object set (or agent set),

7. list of objects

8. size of attribute set (or concept set),

9. list of attributes

10. *(optional)* size of corresponding attributes,

11. *(optional)* list of corresponding attributes.

There are as many such descriptions as there are nodes.

4 Compilation issues

The `networks.tb` suite has initially been developed under Mac OS X, using the XCode library provided by Apple as well as GTK libraries provided through Fink. It should be straightforward to compile the suite using the original `Makefile`. Compilation and execution works also under Linux (kernel versions 2.4 or higher), with a slight modification which is specified in the `Makefile`. Although untested, other UN*Xes should work with similar kinds of modification.

[7] Its value is 1 in case of the existence of a correspondance table. See also points 10 and 11 in the list below.
[8] Through a value function defined in `galois.c`
[9] For details on this variable, see [5] as well as the source (where it is defined by `fortuitude`).

References

[1] C. Lindig. Concepts, a free and portable implementation of concept analysis in C. Open source software package available on http://www.st.cs.uni-sb.de/~lindig/src/concepts-0.3f.tar.gz, 1998.

[2] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 161(6):161–179, 1995.

[3] C. Roth. Co-evolution in epistemic networks – reconstructing social complex systems. *Structure and Dynamics: eJournal of Anthropological and Related Sciences*, 1(3), 2006. (published PhD thesis, Ecole Polytechnique, Paris, France).

[4] C. Roth and P. Bourgine. Epistemic communities: Description and hierarchic categorization. *Mathematical Population Studies*, 12(2):107–130, 2005.

[5] C. Roth and P. Bourgine. Lattice-based dynamic and overlapping taxonomies: the case of epistemic communities. *Scientometrics*, 69(2), 2006.

[6] R. Wille. Concept lattices and conceptual knowledge systems. *Computers Mathematics and Applications*, 23:493, 1992.