Introduction into bipartite networks with Python

Networks Seminar at Karl Franzens University of Graz, Peter Csermely

Stefan Kasberger (1011416)

Graz, on February 10, 2014

1 Introduction

This essay and the related computation delivers a comprehensive introduction into the concept of bipartite networks, a class of networks whose nodes are divided into two sets and only the connection between two nodes in different sets is allowed (Easley and Kleinberg, 2010). The analysis and visualization is done in the programming language Python and offers easy to understand first steps in both fields, network analyses and python programming. As data a collaboration network of github users and projects and an affiliation network of dbpedia entities with countries from KONECT are used. The analysis compares key measures like average shortest path, density, degree centrality and clustering. Also the topic of bipartite network projections to unipartite ones will be addressed on a practical and theoretical level. A specialty is, that the whole documentation and computation is done the open way in use of the Open Source Softwares LaTeX, Python and Git/GitHub to make it reproducible, freely accessible and useable for everyone.

2 Theory

The theoretical basis for networks is the graph theory. By definition, a graph is a way of specifying relationships among a collection of items. It consists of a set of objects, called nodes, with certain pairs of these objects connected by links called edges (Easley and Kleinberg, 2010). Because of this flexible formalism, it is easy to find networks in many domains, like social sciences, micro-biology, psychology and engineering.

Different types of networks exist, depending on their structure and properties. Edges can be directed, from one node to another with a certain direction, or undirected, which means the relation is working in both directions and is symmetric. Typical undirected networks are links on webpages. Friendships for example are mostly undireced. The edges also can be weighted to give the relation a quantifiable dimension, like how often you met each other before or how big the bandwidth of a transmission channel is.

This article focuses on bipartite networks: A particular class of networks, whose nodes are divided into two sets, X and Y, and only the connection between two nodes in different sets is allowed (Easley and Kleinberg, 2010). Nodes from set X are only connected with nodes from set Y, not with other nodes from X, and vice versa. The connections can be weighted/unweighted or directed/undirected.

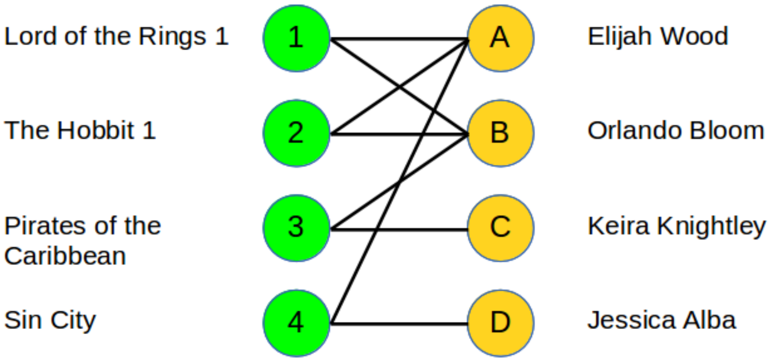


Figure 1: Actor-movie bipartite network. Green nodes are movie nodes, yellow ones are actor nodes.

In order to show the direct relations among a particular set of nodes (Zhou et al., 2007), because of the lack of appropriate notions and tools to address bipartite networks and to compare a particular network to others, one generally transforms a bipartite network into its X- or Y-projection (Latapy, Magnien, and Vecchio, 2008), only consisting of one type of nodes. This is shown best through an example. Let's take a bipartite network of actors (yellow) and movies (green) as shown in figure 1, which gets projected to an actor-actor unipartite network (figure 2). Actor-actor means, that the projection looks on the relations between actors through their connection to movies. When actor x_1 played with actor x_2, the nodes x_1 and x_2 get connected in the projected unipartite graph. This goes on as long as all actor-movie-actor paths were projected. The created unipartite network only exists of actors connected together, and the edges carry the information that the actors played together in a movie.

The projection, more precisely the information loss through it, leads to some serious problems. Some will be mentioned here: 1) It's lost if the actors played in more than one movie together, 2) the bipartite clustering coefficient differs from its unipartite counterpart and varies widely for given unipartite values, demonstrating the distinction between unipartite and bipartite embeddedness in the network (Piepenbrink and Gaur, 2013) and 3) in which movie(s) they played together and. The first problem is solved by the use of a weighted projection, where the resulting network carries the number of co-occurances as edge weight and the second can be tackled by looking at the clustering coefficient of the bipartite network. The last one is an unsolveable part of projections.

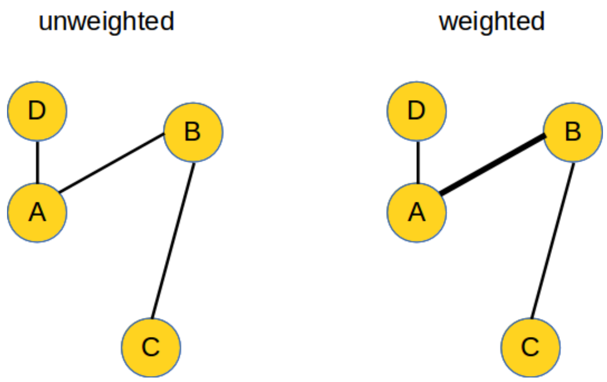


Figure 2: Projected unipartite network from bipartite Actor-Movie network

Another blemish is, that the information contained by the edge whose adjacent X node (Y node) is of degree one, will be lost in X-projection (Y-projection) (Zhou et al., 2007). In sense of computation, the inflationary growth of edges through projection can cause troubles. Notice that each top node of degree d induces links in the X-projection, and conversely (Latapy, Magnien, and Vecchio, 2008). And empiricaly there are also issues: When the number of connections grow, the amount of interactions between the nodes normally decrease, but this information is lost through the projection. Generally this means, that the projection to unipartite networks ignores the possibility that bipartite networks have characteristics unique to their specific nature, which cannot be captured from a unipartite network perspective (Piepenbrink and Gaur, 2013).

There are also specificities of the measurements. It is shown that the expected clustering coefficient in the projections is large, and give an efficient estimation formula; this means that a high clustering coefficient in a projection may be seen as a consequence of the underlying bipartite structure rather than a specific property of the network. Conversely, if the clustering coefficient of the projection is different from the expected one, it means that the underlying bipartite structure has nontrivial properties responsible for it (Latapy, Magnien, and Vecchio, 2008; Newman, Strogatz, and Watts, 2001; Guillaume and Latapy, 2004; Guillaume and Latapy, 2006).

In particular, there is a high heterogenity between degrees of nodes of at least one kind, and there are significant overlaps between neighbourhoods (Latapy, Magnien, and Vecchio, 2008).

3 Analysis and Visualization

3.1 Data

The data used in this introduction is provided by the KONECT[1] project from the Web Science and Technology Institute (WeST)[2] at the University of Koblenz-Landau.

[1] http://konect.uni-koblenz.de/
[2] http://west.uni-koblenz.de/

- dbPedia (*Countries network dataset – KONECT* 2014): Relations between dbPedia[3] entities with countries, like Mozart lived in Austria.
- GitHub (*Github network dataset – KONECT* 2014): Collaboration-network of users and the projects they contributed to.

All KONECT data are licensed under a Creative Commons Attribution-ShareAlike 2.0 Germany License. [4]

3.2 Reproducible Setup and Workflow

To be able to make the analysis reproducible, the process will be done openly, what is summarized under the term Open Science[5]. Only Open Source Software[6] were used together with Open Data Formats[7]. The whole computation and visualization is done with Python's[8] iPython Notebook (Pérez and Granger, 2007), an interactive scientific computing system in which source code and documentation can be written in one single document. Inside it, for the network analysis the module networkX[9] is used and for plotting matplotlib (Hunter, 2007). The iPython notebook is stored in a JSON[10] file and can be exported easily in different formats like Markdown[11], PDF, HTML or TeX[12]. To make the results easily accessible on the internet, understandable and reproducible, everything is written in LaTeX[13] with use of the TU Graz template[14], versioned with Git[15] and put online with the web service GitHub[16]. This work is availiable at `https://github.com/skasberger/se-networks`.

3.3 Preprocessing

The KONECT data is well structured and has not a lot of additional attributes, so there isn't much preprocessing of the data itself necessary. But, because of computational reasons, both networks are way to big to get calculate (shortest path) or the projected in a reasonable amount of time and some calculations need a connected component.

[3] http://dbpedia.org/About

[4] https://creativecommons.org/licenses/by-sa/2.0/de/deed.en

[5] https://en.wikipedia.org/wiki/Open_science

[6] https://en.wikipedia.org/wiki/Open-source_software

[7] https://en.wikipedia.org/wiki/Open_format

[8] http://www.python.org/

[9] http://networkx.github.io/

[10] http://www.json.org/

[11] https://en.wikipedia.org/wiki/Markdown

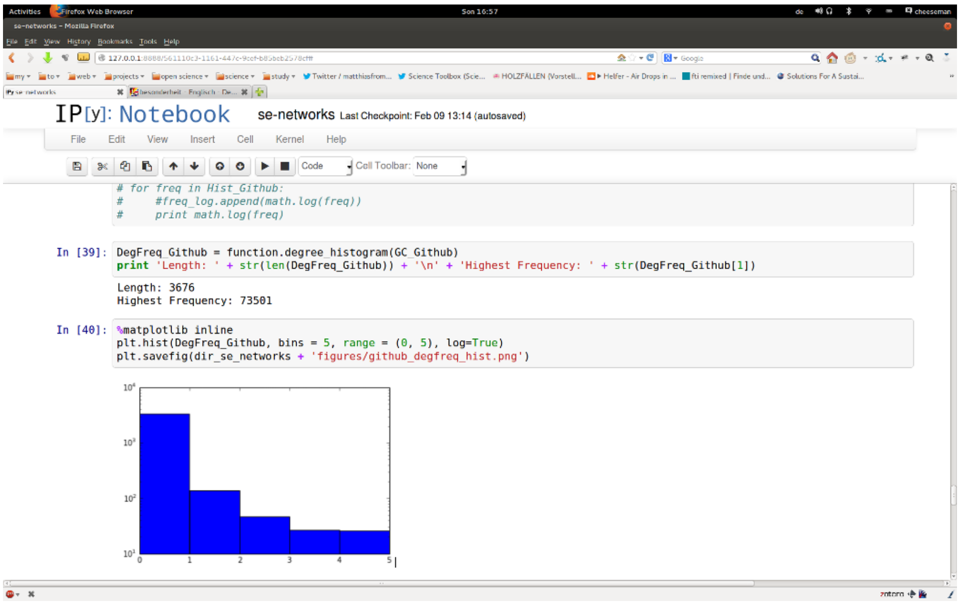
[12] https://en.wikipedia.org/wiki/TeX

[13] http://www.latex-project.org/

[14] http://latex.tugraz.at/vorlagen/allgemein

[15] http://git-scm.com/

[16] https://github.com/



Activities
Firefox Web Browser
Son 16:57
cheeseman
se-networks – Mozilla Firefox
File Edit View History Bookmarks Tools Help
127.0.0.1:8888/561110c3-1161-447c-9cef-b85beb2578cfff
IP[y]: Notebook
se-networks Last Checkpoint: Feb 09 13:14 (autosaved)
File Edit View Insert Cell Kernel Help
Code
Cell Toolbar: None
10^4
10^3
10^2
10^1
0
1
2
3
4
5

```
# for freq in Hist_Github:
#     #freq_log.append(math.log(freq))
#     print math.log(freq)

In [39]: DegFreq_Github = function.degree_histogram(GC_Github)
         print 'Length: ' + str(len(DegFreq_Github)) + '\n' + 'Highest Frequency: ' + str(DegFreq_Github[1])

         Length: 3676
         Highest Frequency: 73501

In [40]: %matplotlib inline
         plt.hist(DegFreq_Github, bins = 5, range = (0, 5), log=True)
         plt.savefig(dir_se_networks + 'figures/github_degfreq_hist.png')
```

Figure 3: Browser Screenshot from iPython notebook environment

To solve these problems the giant component (largest connected component = lcc) was used for expensive calculations. Sometimes even the second largest connected component was used (2nd lcc), because the giant component still was too big (e. g. shortest path, betwenness centrality, closeness centrality). As table 1 and 2 show, the subgraphs (components) differ a lot in structure and key metrics from the parent graph. It is important to distinguish in the analysis between which graph was used to make right conclusion.

Graph	Users	Projects	Nodes	Edges	Avg. Degree
Parent Graph	56.519	120.867	177.386	440.237	4,96
Giant Component	39.845	99.907	139.752	417.361	5,97
2nd lcc	3	42	544.947	580.231	2,12

Table 1: Properties of GitHub graphs

Graph	Entitites	Countries	Nodes	Edges	Avg. Degree
Parent graph	548.077	2.445	550.522	584.947	2,12
Giant component	543.589	1.358	544.947	580.231	2,12
2nd lcc	337	1	338	337	1,99

Table 2: Properties of dbpedia graphs

The difference of key metrics continues and intensifies through the projection, as the strong increase of edges and the average degree (see table 3) in the dbpedia graph show. For the github graph it was a user-user projection, for the dbpedia an entitie-entitie projection of the 2nd lcc graph. The high degree of dbpedia lies in the very dense and easy structure of only one country connected to 338 entities in the bipartite graph.

Graph	Nodes	Edges	Avg. Degree
Projected GitHub	3	3	2,00
Projected dbpedia	337	56.616	336,00

Table 3: Properties of projected graphs

As last step, the graphs are tested if the components are connected and if isolated nodes exist.

3.4 Degree

Degree distribution is one of the most important centrality measurements. The whole graphs of both data sets were used for the calculation. As result, the degree distribution was more dense in the low area of the dbpedia network as in Github (see figure 4). This shows, that the average contribution on github is higher than the average number of connections from dbpedia entities to countries, what sounds fair right.

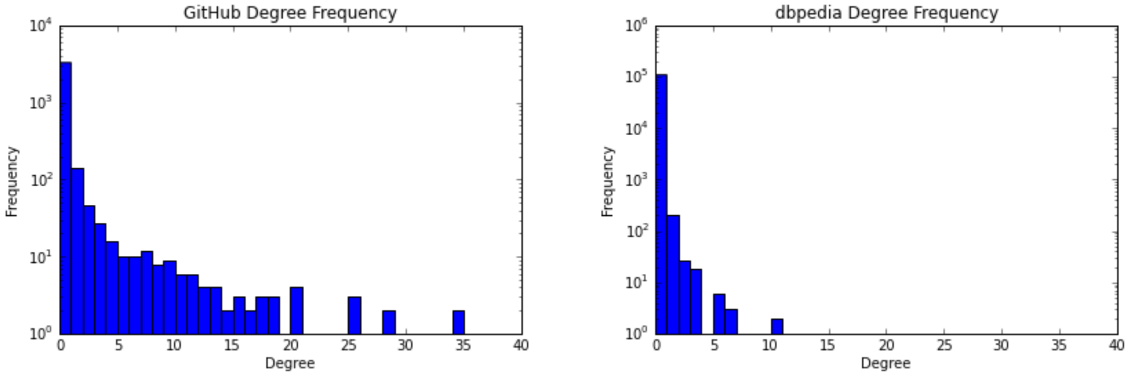


Figure 4: Degree distribution on a log-lin scale. The left shows the github network, on the right is dbpedia

3.5 Shortest Path

Shortest Path describes the shortest possible way between two nodes. To get the average shortest path of a graph, every possible connection between all nodes have to be computed and divided by the number of paths. Because of the high computational costs for this, the second largest connected components are used. As a result, the github network has slightly higher average shortest path (2,11) than dbpedia (1,99).

3.6 Clustering

Clustering tells about the density of the graph. This is a very imporant measure for bipartite networks and their projections (Latapy, Magnien, and Vecchio, 2008). For the analysis the 2nd lcc network was used, because of computational costs. The average clustering coefficient of the simple dbpedia graph with only one country shows the expected high clustering of the related entities. In general, the clustering coefficients of a projected graph should be much higher than of the original bipartite one. In this case, due to of the very untypical structure of the bipartite networks, this effect can not be seen.

Graph	Avg. clustering coefficient
Github all	0,90
Github users	0,18
Github projects	0,95
Github projected	1
dbpedia all	1,0
dbpedia entities	1.00
dbpedia countries	0.00
dbpedia projected	1

Table 4: Clustering coefficients

3.7 Diameter

Diameter is the longest of all shortest paths in a network. Because of the underlying cost expensive shortest path algorithm, again, the 2nd lcc network was used. The github network has a diameter of 4, the dbpedia one of 2.

4 Conclusion

The study of networks is still in it's beginning, and so is the research on bipartite networks. A lot of ideas come up for research on 1) weighting algorithms of projections, 2) specialization in the needs of visualization (Schulz et al., 2008) and 3) new measurements specifically for bipartite networks and further research on the existing ones. To make knowledge easier transferable, comparability of studies across disciplines and fields should be strived for (Piepenbrink and Gaur, 2013).

The impression of the first work with iPython was very good. Together with matplotlib it makes a documented, sequencial, integrated and open workflow easily possible. The computation itself proved to be challenging. Networkx is easy to use but the drawing functions are very scarse and need more specific layouts. iGraph looks like a good alternative for this. Also some issues with algorithms occured, maybe failure on my side, maybe bugs. Another problem was typical for large-scale real world networks: the computational costs were huge and not manageable by normal IT-infrastructure, like a standard laptop. For this parallel computing would be necessary.

In the analysis, much more in detail could be done: Comparing the measurements of the two node sets with one another, or comparing it with random graphs for example.

All in all, many questions arose for further investigations on bipartite networks, which would be very helpful to understand network theory in general and the specifics of bipartite networks in particular.

5 Openness

5.1 Copyright

This work is licensed under the Creative Commons Attribution 3.0 AT license[17].

All generated code, content and figures are online freely available on the se-networks GitHub repository[18] and compatible with the OpenDefinition[19]. The sourcecode is licensed under the MIT license[20], figures and text under the Creative Commons Attribution 3.0 AT license.

5.2 openscienceASAP

An own webpage[21] was created for the Networks Seminar at openscienceASAP[22] to collect all informations and works on one central page.

[17] https://creativecommons.org/licenses/by/3.0/at/

[18] https://github.com/skasberger/se-networks

[19] http://opendefinition.org/

[20] http://opensource.org/licenses/MIT

[21] http://opensicenceasap.org/education/courses/se-networks/

[22] http://openscienceasap.org

5.3 Other Sources

- bipartite network dbPedia from KONECT (CC BY-SA) `http://konect.uni-koblenz.de/networks/dbpedia-country` (*Countries network dataset – KONECT* 2014)
- bipartite network GitHub from KONECT (CC BY-SA) `http://konect.uni-koblenz.de/networks/github` (*Github network dataset – KONECT* 2014)

6 Bibliography

References

Countries network dataset – KONECT (Jan. 2014). URL: http://konect.uni-koblenz.de/networks/dbpedia-country.

Easley, David and Jon Kleinberg (2010). *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. New York, NY, USA: Cambridge University Press. ISBN: 0521195330, 9780521195331.

Github network dataset – KONECT (Jan. 2014). URL: http://konect.uni-koblenz.de/networks/github.

Guillaume, Jean-Loup and Matthieu Latapy (June 15, 2004). "Bipartite structure of all complex networks". In: *Information Processing Letters* 90.5. 00150, pp. 215–221. ISSN: 0020-0190. DOI: 10.1016/j.ipl.2004.03.007.

– (Nov. 15, 2006). "Bipartite graphs as models of complex networks". In: *Physica A: Statistical Mechanics and its Applications* 371.2. 00099, pp. 795–813. ISSN: 0378-4371. DOI: 10.1016/j.physa.2006.04.047.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3, pp. 90–95.

Latapy, Matthieu, Clémence Magnien, and Nathalie Vecchio (2008). "Basic notions for the analysis of large two-mode networks". In: *Social Networks* 30.1, pp. 31–48. ISSN: 03788733. DOI: 10.1016/j.socnet.2007.04.006.

Newman, M. E. J., S. H. Strogatz, and D. J. Watts (July 2001). "Random graphs with arbitrary degree distributions and their applications". In: *Physical Review E* 64.2. 02317 arXiv:cond-mat/0007235. ISSN: 1063-651X, 1095-3787. DOI: 10.1103/PhysRevE.64.026118. URL: http://arxiv.org/abs/cond-mat/0007235 (visited on 02/09/2014).

Pérez, Fernando and Brian E. Granger (May 2007). "IPython: a System for Interactive Scientific Computing". In: *Computing in Science and Engineering* 9.3, pp. 21–29. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.53. URL: http://ipython.org.

Piepenbrink, Anke and Ajai Gaur (2013). *Methodological Advances in the Analysis of Bipartite Networks: An Illustration Using Board Interlocks in Indian Firms*. SSRN Scholarly Paper ID 2199111. Rochester, NY: Social Science Research Network. URL: http://papers.ssrn.com/abstract=2199111.

Schulz, Hans-Jörg et al. (2008). "Visual Analysis of Bipartite Biological Networks". In: *Proceedings of the First Eurographics Conference on Visual Computing for Biomedicine*. EG VCBM'08. Aire-la-Ville, Switzerland: Eurographics Association, 135–142. ISBN: 978-3-905674-13-2. DOI: 10.2312/VCBM/VCBM08/135-142.

Zhou, Tao et al. (2007). "How to project a bipartite network?" In: *Physical Review E* 76.4. arXiv:0707.0540 [physics]. ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.76.046115. URL: http://arxiv.org/abs/0707.0540.