Proximity-based Methods for Link Prediction in Graphs with R package linkprediction

Michał Bojanowski
Kozminski University

Bartosz Chroł
ICM, University of Warsaw

Abstract

Link prediction is a problem of predicting future edges of an undirected graph based on a single snapshot data of that graph. Vertex proximity measures are indicies giving numerical scores for every pair of vertices in a graph that can be used for predicting future edges. This short note describes an R package **linkprediction** implementing 20 different vertex similarity and proximity measures from the literature. The article provides the definitions of implemented measures, describes the main user-facing functions, and illustrates the use of the methods with a problem of predicting future co-authorship relations between researchers of University of Warsaw.

Keywords: network analysis, link prediction, R.

1. Introduction

Graphs are a popular way of representing structures of interactions between elements of a studied system. For example, in social sciences the vertices are used to represent people, organizations or other social entitites while edges represent ties such as friendship, collaboration or flow of capital. As such, network data and methods of its analysis appear in many disciplines (Wasserman and Faust 1994; Brandes, Robins, McCranie, and Wasserman 2013; Barabási 2016; Jackson 2010; Goyal 2012; Newman 2010).

In many settings networks are dynamic (for example Van de Bunt, Van Duijn, and Snijders 1999; Ferligoj, Kronegger, Mali, Snijders, and Doreian 2015, and many others). The question arises whether and how we can use network data about the past or present to formulate predictions about the topology of the network in the future. Modeling evolution of graphs is a complex problem with an ongoing research effort in formulating statistical models. One example of such model is Temporal Exponential-family Random Graph Model (TERGM, Hanneke, Fu, and Xing 2010; Krivitsky and Handcock 2014) in which time is assumed to be discrete and the dynamics of the graph is represented by two conditional probability distributions – for edge formation and edge dissolution – each specifed as an Exponential-family Random Graph Model. Another example is the Stochastic Actor-Oriented Model (SAOM, Snijders 1996) in which network change is modeled as a continuous time process of actors (vertices) making multinomial choices about forming or dissolving network edges they are incident on. Estimation of mentioned models require that we observe the network at least at two points in time. Additionaly, because the estimation relies on Markov Chain Monte Carlo methods (see R packages **tergm** Krivitsky and Handcock 2018, and **RSiena** Ripley, Snijders, Boda, Vörös, and Preciado (2018)), it becomes hardware-challenging to fit these models to network data beyond a couple thousand vertices. In both cases fitted models can be used to simulate future realizations of the graph.

A related and somewhat simpler problem is trying to predict future edges based on a single snapshot of the network. This problem has been framed as a problem of *link prediction* (Liben-Nowell and Kleinberg 2007). Approaches to solve it include various *node similarity* or *proximity-based* indices. These indices allow for computing a score for every pair of vertices in the given network which in turn can be used for predicting if an edge is likely in the future. The indicies usually implement heuristics and qualitative ideas about how the network evolves. In contrast to statistical modeling frameworks mentioned above, the node proximity measures are, on the one hand, rather simplistic and provide very limited insights about "how" or "why" the network evolves. On the other hand, they are usually computationally much less demanding which makes their application to larger datasets more feasible. Proximity indices for link prediction have been used for example to: identify proteins likely to interact (Clauset, Moore, and Newman 2008), prototype recommendation engines on social networking websites (Backstrom and Leskovec 2011), predict future co-authroship links (Liben-Nowell and Kleinberg 2007), or forecast dynamics of terrorist networks (Desmarais and Cranmer 2013).

In this short note we present an R package **linkprediction** (Bojanowski and Chrol 2018) that provides implementations of 20 node proximity indices collected through an extensive review of the literature. To our knowledge these methods are not available to R community apart from the function `similarity()` in the **igraph** package (Csardi and Nepusz 2006) implementing three of these indices. Other features of the presented **linkprediction** package are:

- It supports objects of class "igraph" (package **igraph**, Csardi and Nepusz 2006) or "network" (package **network**, Butts 2008, 2015) – the two probably most popular classes for network data in R.
- Where possible, functions use sparse matrices for efficient computation.
- For every index the results can be returned in three forms: a matrix, an edgelist data frame, or an "igraph" object with the scores assigned as edge attributes. This facilitates further analysis with functions from other packages. We provide more details and illustrations in Section 4.
- An example dataset with a subgraph of a co-authorship network from University of Warsaw (1486 vertices, 7505 edges) is provided to facilitate examples and possibly testing new measures/approaches to link prediction.

The remainder of the article is organized as follows. In Section 2 we provide a formal definition of a node proximity index. Section 3 provides a detailed list of implemented methods. Section 4 showcases the `proxfun()` function – the main interface to all the methods – its arguments and types of values that it can return. The article is concluded with a short illustrative example of predicting co-authorship links among the researchers of University of Warsaw in Section 5.

2. Generic node proximity index

To introduce some notation let *graph* G consist of a set of *vertices* V and a set of *edges* $E \subseteq V \times V$ between these vertices. We will interchangably use terms "nodes" and "links" for "vertices" and "edges" respectively. A pair of vertices is called *a dyad*. If there exists an edge between two vertices, they are adjacent to each other. An *adjacency matrix* A is a matrix representation of a graph. It is a square matrix with generic element a_{xy} equal 1 if vertices x and y are adjacent and 0 otherwise.

A node proximity index is a function S giving a real number score to every dyad in graph G:

$$S(x, y) : V \times V \mapsto \Re \tag{1}$$

It is convinent to arrange values of $S(x, y)$ into a matrix $[s_{xy}]$. We will use the shorthand s_{xy} in the definitions of various measures in Section 3.

The terms "node similarity index" or "proximity-based index" seem to be used rather interchangeably in the literature. For the sake of clarity we will use the first of the two terms throughout this article. Mentioned terms may also be a source of confusion because of two related concepts in network analysis: homophily and graph distance.

First, homophily is one of the mechanisms often found important in explaining structure of networks (McPherson, Smith-Lovin, and Cook 2001). It implies that network edges tend to be more likely between vertices *similar* to each other in terms of the specified vertex attribute such as gender, age, taste in music etc. (see also Bojanowski and Corten 2014). In contrast to the this homophily-related understanding, "node similarity" indices covered in this article do not use any information about possible vertex attributes and the term "similarity" has a rather informal meaning.

Second, graph distance is an important concept in graph theory. It is defined as the length of the shortest path connecting two vertices in the graph. Two vertices are said to be proximate if the graph distance between them is relatively short. Most of the indices we cover below use that concept more or less directly.

3. Overview of implemented methods

The methods implemented in the package and described in this section have been gathered from many different sources. All measures give proximity score between two vertices. Some measures are symmetrical by definition, some had to be modified to achieve symmetry. The scores of different measures have different scales, but for link prediction the rankings of dyads according to scores are of primary importance.

Following Lü and Zhou (2011), we group proximity measures into three categories: local, quasi-local, and global. Local methods focus only on the properties of neighborhoods of the given pair of vertices. Global methods take into account the information about the network as a whole. Quasi-local methods lie somewhere in between the local and global methods. They need more information than local methods, but still do not need information about the whole graph.

In the presented **linkprediction** package measures can be computed with the function `proxfun()`, which we will describe in more detail in Section 4. The descriptions of the measures in the following sections contain short strings in parentheses next to the measure name. These are names or acronyms that can be supplied to the `method` argument of `proxfun()` function to select the appropriate measure, for example the call `proxfun(g, method="aa")` will compute

Symbol	Description
$\lvert Q\rvert$	Cardinality of some set Q,
$A = [a_{xy}]$	Adjacency matrix of a graph,
n	Number of vertices in the graph,
x, y, z	Generic vertices,
$\Gamma(x)$	Set of all neighboring vertices of vertex x,
$k_x = \sum_j a_{xj} = \lvert\Gamma(x)\rvert$	Degree of vertex x,
$paths_{xy}^{<l>}$	Set of all paths of length l from x to y,
$D = \begin{bmatrix} k_1 & 0 & \dots & 0 \\ 0 & k_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \dots & k_n \end{bmatrix}$	Degree matrix,
$L = D - A$	Laplacian matrix,
$S = [s_{xy}]$	Proximity matrix,
L^+	Moore-Penrose pseudo-inverse of matrix L.

Table 1: Summary of notation.

Adamic-Adar proximity scores for all dyads in supplied graph `g`.

3.1. Additional notation

We will use the following additional notation in measure definitions. A *subgraph* G' of a graph G is a graph with vertices $V' \subseteq V$ and edges $E' = E \cap V' \times V'$. A *path* connecting vertices x and y is a series of adjacent edges starting from x and finishing in y. A graph is *connected* if every pair of vertices is connected by a path, otherwise it is *disconnected*. The largest connected subgraph of a graph is called a *giant component*. A *neighborhood* of the vertex x is a set of vertices adjacent to x. The remaining notation is presented in Table 1. All vectors are assumed to be column vectors.

3.2. Local methods

Most of the local measures are variations of the "common neighbors" measure (Newman 2001).

Common neighbors (`cn`) (Newman 2001) The measure implements an intuition that two scientists are more likely to collaborate if they have collaborated with the same group of people in the past. Newman (2001) used this method in the study of collaboration networks, showing positive relation between the number of common neighbors and probability of collaborating in the future.

$$s_{xy} = \lvert\Gamma(x) \cap \Gamma(y)\rvert, \tag{2}$$

Salton Index (`cos`) It measures the cosine of the angle between columns of the adjacency matrix, corresponding to given vertices. This measure is commonly used in information retrieval.

$$s_{xy} = \frac{\lvert\Gamma(x) \cap \Gamma(y)\rvert}{\sqrt{k_x \times k_y}}. \tag{3}$$

Jaccard Index (`jaccard`) (Jaccard 1912) Jaccard Index measures how many neighbors of given nodes are shared. It reaches its maximum if $\Gamma(x) = \Gamma(y)$, that means all neighbors are shared.

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}. \tag{4}$$

Sørensen Index (`sor`) (Sørensen 1948) This method is similar to Jaccard Index, as it measures a relative size of an intersection of neighbors' sets.

$$s_{xy} = \frac{2|\Gamma(x) \cap \Gamma(y)|}{k_x + k_y}. \tag{5}$$

Hub Promoted Index (`hpi`) (Ravasz, Somera, Mongru, Oltvai, and Barabási 2002) This measure assigns higher scores to links adjacent to hubs (high-degree nodes), as the denominator depends on the lower degree only.

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\min\{k_x, k_y\}}. \tag{6}$$

Hub Depressed Index (`hdi`) (Ravasz *et al.* 2002) This measure, in contrast to Hub Promoted Index, assigns lower scores to links adjacent to hubs, since it penalizes big neighborhoods.

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{\max\{k_x, k_y\}}. \tag{7}$$

Leicht-Holme-Newman Index (`lhn_local`) (Leicht, Holme, and Newman 2006) A variant of Common Neighbors, similar to Salton Index

$$s_{xy} = \frac{|\Gamma(x) \cap \Gamma(y)|}{k_x \times k_y}. \tag{8}$$

Preferential Attachment (`pa`) (Barabási and Albert 1999) Preferential Attachment was developed as a model for the growth of a network in the sense of arrival of new nodes. This is often not really the case as the network might evolve (new links are added and some existing removed) without changes to the node set. However, if we follow the intuition behind the preferential attachment model, namely nodes with high degree are more attractive to connect to, we may expect that links are more likely to be incident on nodes with high degree. Hence:

$$s_{xy} = k_x \times k_y. \tag{9}$$

Adamic-Adar Index (`aa`) (Adamic and Adar 2001) This measure extends the idea of counting common neighbors by introducing weights inversely proportional to their degrees. A common neighbor, which is unique to only a few nodes, is more important (has more weight) than a high-degree node. Note that if a node z is a common neighbor of nodes x and y, than its degree is at least 2.

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k_z}. \tag{10}$$

Resource Allocation Index (`ra`) (Zhou, Lü, and Zhang 2009) This measure is motivated by a resource transmission process in which common neighbors of nodes x and y play a role of transmitters spreading a unit of a resource. With an additional assumption that each transmitter spreads its resource equally across links the measure captures how much resources y received from x (or vice versa) (c.f. Section 5 of Zhou *et al.* 2009).

$$s_{xy} = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k_z}. \tag{11}$$

3.3. Global methods

Here we describe global methods.

Katz Index (`katz`) (Katz 1953) Katz Index counts all the paths between the given pair of nodes, with shorter paths having larger weights.

$$s_{xy} = \sum_{l=1}^{\infty} \beta^l |paths_{xy}^{<l>}|, \tag{12}$$

where β is a free parameter. The sum converges when β is lower than the reciprocal of the largest eigenvalue of adjacency matrix. If this condition is satisfied Katz Index can be expressed in a matrix form

$$S = (I - \beta A)^{-1} - I. \tag{13}$$

where A is the adjacency matrix and I is the identity matrix.

Leicht-Holme-Newman Index, global version (`lhi_global`) (Leicht *et al.* 2006) This is a variant of Katz Index, based on the concept that two nodes are proximate if their neighbors are proximate themselves. It counts all paths between two nodes, but weights them by the expected number of such paths in a random graph with the same degree distribution. This measure is proportional to the following matrix expression:

$$S = D^{-1} \left(I - \frac{\phi A}{\lambda_1} \right)^{-1} D^{-1}, \tag{14}$$

where λ_1 is the largest eigenvalue of adjacency matrix A and ϕ is a free parameter.

Average Commute Time (`act`) (Klein and Randić 1993) ACT similarity index is given by

$$s_{xy} = \frac{1}{n(x,y)} = \frac{1}{m(x,y) + m(y,x)}, \tag{15}$$

where $m(x,y)$ is the average number of steps required by a random walker starting from x to reach y. To achieve symmetry we take the sum of two directional commute times. Thus, two nodes are similar if they are closer to each other and have shorter commute time. Average Commute Time could be computed by solving a collection of linear equations stemming from a Markov Chain analysis, but it is more straightforward to compute it in terms of the pseudo-inverse of the Laplacian matrix, L^+. Namely:

$$n(x,y) = 2M(l_{xx}^+ + l_{yy}^+ - 2l_{xy}^+), \tag{16}$$

where $l^+_{xy} = [L^+]_{xy}$ and M is the number of edges. Thanks to the special form of the Laplacian matrix, its pseudoinverse L^+ could be computed using the formula of Fouss, Pirotte, Renders, and Saerens (2007):

$$L^+ = \left(L - \frac{ee^T}{n}\right)^{-1} + \frac{ee^T}{n}, \tag{17}$$

where e is a column vector made of 1s.

Normalized Average Commute Time (`act_n`) (Klein and Randić 1993) is a variant of ACT above, which takes into account node degrees, as for high-degree node (hub) y, $m(x, y)$ is usually small regardless of x.

$$s_{xy} = \frac{1}{(m(x,y)\pi_y + m(y,x)\pi_x)}, \tag{18}$$

where π is a stationary distribution of a Markov chain describing random walker on the graph. It can be shown that on a connected graph

$$\pi(x) = \frac{k_x}{\sum_y k_y}. \tag{19}$$

Cosine based on L^+ (`cos_l`) (Fouss *et al.* 2007) It measures the cosine of the angle between node vectors in a space spanned by columns of L^+.

$$s_{xy} = \frac{l^+_{xy}}{\sqrt{l^+_{xx} l^+_{yy}}} \tag{20}$$

Random Walk with Restart (RWR) This is an adaptation of the PageRank algorithm (Brin and Page 1998). Consider a random walker starting from node x and periodically, with probability α, returning to x. Let q_x be a stationary distribution of a Markov chain describing this walker. From definition of stationary distribution:

$$q_x = (1-\alpha)P^T q_x + \alpha e_x, \tag{21}$$

where e_x is a unit vector with 1 on position corresponding to node x, and P is a transition matrix describing ordinary random walker, $P_{xy} = 1/k_x$ if $A_{xy} = 1$ and 0 otherwise. The solution for all nodes simultaneously is

$$q = [q_1|q_2|\ldots|q_n] = \alpha(I - (1-\alpha)P^T)^{-1}. \tag{22}$$

In order to achieve symmetry the RWR index is defined as

$$s_{xy} = q_{xy} + q_{yx}. \tag{23}$$

L^+ directly (`l`) (Fouss *et al.* 2007) L^+ provides a direct measure of proximity, as its elements are the inner products of vectors from an Euclidean space, which preserves Average Commute Time between nodes (see Fouss *et al.* 2007, for details).

$$S = L^+. \tag{24}$$

Matrix Forest Index (`mfi`) (Chebotarev and Shamis 1997) Matrix Forest Index can be understood as the ratio of (1) the number of spanning rooted forests such that nodes x and y belong to the same tree rooted at x to (2) the number of all spanning rooted forests of the network. See Chebotarev and Shamis (1997) for detailed derivations.

$$S = (I + L)^{-1}. \tag{25}$$

3.4. Quasi-local methods

Geodesic distance In this approach we expect edges to appear more likely between the vertices that are closer to each other in terms of geodesic distance. The proximity index becomes:

$$s_{xy} = \begin{cases} \infty & \text{if } x = y \\ 0 & \text{if } x \text{ and } y \text{ are not connected} \\ \frac{1}{p_{xy}} & \text{in other cases} \end{cases} \tag{26}$$

where $p_{xy} = \min\{l : path_{xy}^{<l>} \text{ exists}\}$ is the length of the shortest path connecting x and y. It is not implemented in **linkprediction** package but available with code{igraph::distances()}. We list it here for the sake of completness.

Local Path Index (`lp`) (Zhou *et al.* 2009)

$$S = A^2 + \epsilon A^3, \tag{27}$$

where ϵ is a free parameter. This measure benefits from more information than simple common neighbors, as it looks at neighborhoods of second order.

4. Usage

The main function in the package is `proxfun()` which calculates scores of selected node proximity measure (argument `method`) based on the provided graph (argument `graph`). Let us use the following simple graph as an example (see Figure 1).

```
R> library(igraph)
R> g <- make_graph( ~ 1 -- 2:3, 4 -- 2:3:5)
```

To calculate scores of, for example, Common Neighbors we call `proxfun()` with argument `method = "cn"`. By default the result is a square matrix of scores. Rows and columns correspond to vertex ids in the graph object.

```
R> proxfun(g, method="cn")

  1 2 3 4 5
1 0 0 0 2 0
2 0 0 2 0 1
3 0 2 0 0 1
4 2 0 0 0 0
5 0 1 1 0 0
```

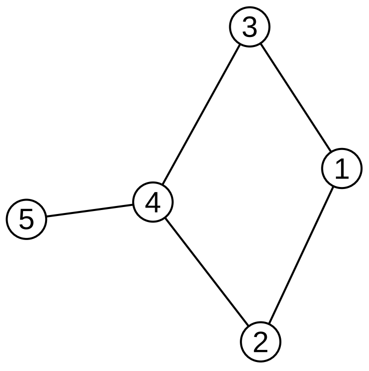

Figure 1: Simple example graph.

The matrix is by definition symmetric. We see can that, for example, vertex 5 has one neighbor in common with vertices 2 and 3 (vertex 4 in both cases) while vertex 2 has two neighbors in common with vertex 3 (vertices 1 and 4) and one neighbor in common with vertex 5 (vertices 4 mentioned earlier).

Alternatively, and this can be controlled with argument `value`, the function can return a data frame containing an edge list (`value="edgelist"`). This is a data frame with columns:

- `from`, `to` – vertex ids of the adjacent vertices
- `value` – scores of the selected proximity index

```
R> proxfun(g, method="cn", value="edgelist")

  from to value
1    4  1     2
2    3  2     2
3    5  2     1
4    2  3     2
5    5  3     1
6    1  4     2
7    2  5     1
8    3  5     1
```

In this form dyads that received score of 0 are removed. Additionally, even though the scores are symmetric, the edge list contains all non-zero score values. This redundancy is deliberate as it facilitates easy joining such a data frame with possibly other dyadic data about the given network.

The third option is `value="graph"`. The object returned is an "igraph" object with the same vertex set as the supplied graph and with edges in all dyads that received a non-zero score. The score itself is stored in an edge attribute `"weight"`. For example:

```
R> g.cn <- proxfun(g, method="cn", value="graph")
R> g.cn

IGRAPH 06af9a4 UNW- 5 4 --
+ attr: name (v/c), weight (e/n)
+ edges from 06af9a4 (vertex names):
[1] 3--5 2--5 2--3 1--4

R> E(g.cn)$weight

[1] 1 1 2 2
```

5. Illustrative example

Let us consider the problem of predicting whether two researchers who did not collaborate in the past will co-author a publication together. We will use the data `uw` provided with the **linkprediction** package.

5.1. Data

Data `uw` provided with the package **linkprediction** is an `igraph` object representing an undirected graph of 1486 researchers (vertices) connected with 7505 edges. Two researchers are connected if they co-authored at least one publication in the period 2007-2012. The graph was assembled from bibliographic data extracted from Polish Scholarly Bibliography (PBN 2017). A publication record was included if at least one of the authors was an employee of University of Warsaw. The network in the `uw` object is a subgraph of that larger data consisting of researchers who (1) published at least once in 2007-2009, (2) published at least once in 2010-2012, and (3) are members of the largest connected component of the co-authorship graph based on publications in 2007-2009.

The network has additional vertex and edge attributes. In particular:

- `affiliation` – Vertex attribute identifying groups of departments by scientific field: natural sciences, social sciences, humanities, other, and external (co-authors who are not employees of UW).
- `p1` and `p2` – Logical edge attributes. If `p1` is `TRUE` then researchers incident on such an edge co-authored at least one publication in the first period (2007-2009). Analogously, if `p2` is `TRUE` then incident researchers co-authored at least one publication in the second period (2010-2012).

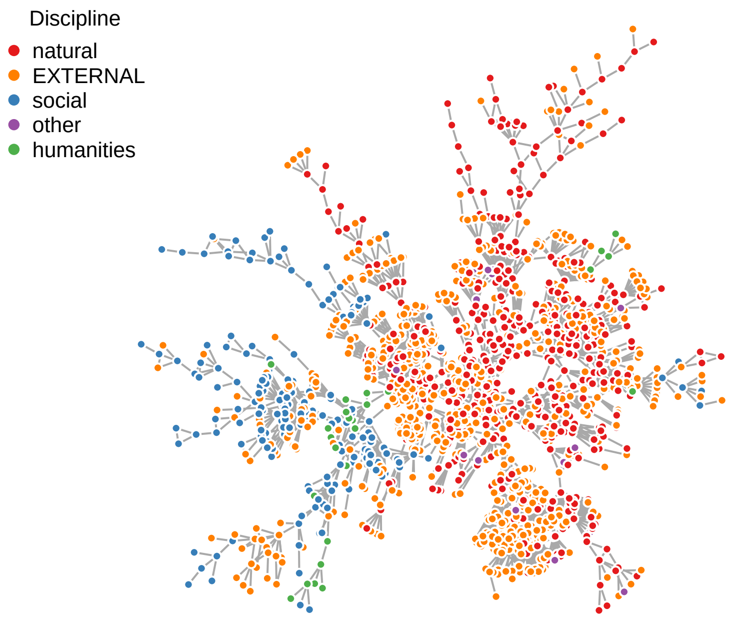

Figure 2: Co-authorship network in period 1.

The network in the first period (i.e. with edges for which `p1` is `FALSE` removed) is shown in Figure 2.

5.2. Analysis

Our goal is to use the co-authorship network from period 1 to predict edges in period 2 in dyads that were disconnected in period 1. These are the *new* co-authorship ties. To this end we will use three of the methods described in detail earlier in Section 3:

- Adamic-Adar Index (`method="aa"`).
- Preferential Attachment (`method="pa"`)
- Cosine based on L^+ (`method="cos_l"`)

The procedure will consist of the following steps:

1. Calculate the scores for the three measures on the network from period 1 (“training data”)
2. Create the “test data” consisting of dyads that were not connected in period 1 labelled with `TRUE` or `FALSE` depending whether there is a tie in that dyad in period 2.
3. Create and evaluate predictions based on the scores from (1) using ROC curves with package **ROCR** (Sing, Sander, Beerenwinkel, and Lengauer 2005).

Let us make the training data as a subgraph of co-authorships from period 1, in other words with period 2 edges removed:

```
R> train <- delete_edges(uw, E(uw)[!p1])
```

Now we calculate the three measures: Adamic-Adar, Preferential Attachment, and Cosine similarity based on L^+. We return the results as data frames (edgelists) to be able to join them with the test data created in the next step. As the graph is undirected we keep only the dyads for which `from` is less than `to`. Finally, the data frame returned by `proxfun()` has column `value`, which we rename with the name of the used proximity measure. For processing the data frames we use functions from packages **dplyr** (Wickham, François, Henry, and Müller 2018) and **tidyr** (Wickham and Henry 2018).

```
R> # Suppressing package conflict messages to save space
R> library(dplyr, warn.conflicts=FALSE)
R> library(tidyr, warn.conflicts=FALSE)
R>
R> aa <- proxfun(train, method="aa", value="edgelist") %>%
R+   filter(from < to) %>%
R+   rename(aa=value)
R> pa <- proxfun(train, method="pa", value="edgelist") %>%
R+   filter(from < to) %>%
R+   rename(pa=value)
R> cosi <- proxfun(train, method="cos_l", value="edgelist") %>%
R+   filter(from < to) %>%
R+   rename(cosi=value)
```

Above gives us three data frames `aa`, `pa`, and `cosi`. The inital rows of data frame `aa` are the following:

```
R> head(aa)
```

```
  from to        aa
1    2  6 0.4808983
2    2  7 0.4808983
3    6  7 1.4921045
4    9 10 0.6378580
5    2 12 0.3034131
6   14 18 3.2663405
```

Let us now prepare the test data. It has all dyads from graph `uw`.

```
R> testdf <- tidyr::crossing(
R+   # All dyads -- all pairs of vertex ids
R+   from = seq(1, vcount(train)),
R+   to = seq(1, vcount(train))
R+ ) %>%
R+   # The network is undirected thus we keep
R+   # only unique unordered pairs of vertex ids
```

```
R+   filter(from < to) %>%
R+   # Join "true" edges from period 2
R+   left_join(
R+    igraph::as_data_frame(uw, what="edges") %>%
R+      as_tibble() %>%
R+      transmute(
R+        from = match(from, V(uw)$name),
R+        to = match(to, V(uw)$name),
R+        p1,
R+        p2
R+      ),
R+    by = c("from", "to")
R+   ) %>%
R+   # Dyads without a match (have NAs) are disconnected
R+   # so we convert NAs to FALSE
R+   mutate_at(
R+     c("p1", "p2"),
R+     function(x) ifelse(is.na(x), FALSE, x)
R+   ) %>%
R+   # Create logical variable `test` to flag new co-authorships.
R+   # These are present in `p2` but absent in `p1`.
R+   mutate(
R+     test = p2 & !p1   # new co-authorships
R+   )
```

At this stage the `testdf` data frame contains all dyads in graph `uw`. Initial rows are:

```
R> head(testdf)
```

```
# A tibble: 6 x 5
   from    to p1    p2    test
  <int> <int> <lgl> <lgl> <lgl>
1     1     2 FALSE FALSE FALSE
2     1     3 FALSE FALSE FALSE
3     1     4 FALSE FALSE FALSE
4     1     5 FALSE FALSE FALSE
5     1     6 FALSE FALSE FALSE
6     1     7 FALSE FALSE FALSE
```

Every dyad is in one of four states. It can be

1. Diconnected in both periods – the two researchers did not work together at all.
2. Disconnected in period 1, but connected in period 2 – the two researchers started collaborating in period 2.
3. Connected in period 1, but disconnected in period 2 – the two researchers stopped collaborating in period 2.

4. Connected in both periods – the two researchers collaborated for the whole time under study.

The frequencies of those four states can be obtained by counting dyads depending on their connectedness in period 1 (variable `p1`), connectedness in period 2 (variable `p2`):

```
R> testdf %>%
R+   count(p1, p2, test)
```

```
# A tibble: 4 x 4
  p1    p2    test        n
  <lgl> <lgl> <lgl>   <int>
1 FALSE FALSE FALSE 1095850
2 FALSE TRUE  TRUE     1343
3 TRUE  FALSE FALSE    2069
4 TRUE  TRUE  FALSE    4093
```

Thus among 1103355 all possible pairs of researchers 2069 + 4093 = 6162 collaborated in period 1 and 1343 + 4093 = 5436 collaborated in period 2, of which 1343 are the *new* collaborations that we want to predict. For these dyads the logical variable `test` is equal to `TRUE`. As mentioned earlier, we limit our predictive task to pairs of authors who did not collaborate in period 1:

```
R> testdf <- testdf %>%
R+   filter(!p1)
```

Let us join the predictions computed earlier:

```
R> preds <- testdf %>%
R+   left_join(aa, by=c("from", "to")) %>%
R+   left_join(pa, by=c("from", "to")) %>%
R+   left_join(cosi, by=c("from", "to")) %>%
R+   mutate_at(
R+     c("aa", "pa", "cosi"),
R+     funs(ifelse(is.na(.), 0, .))
R+   )
```

The result is a data frame with the following initial rows

```
R> head(preds)
```

```
# A tibble: 6 x 8
   from    to p1    p2    test     aa    pa  cosi
  <dbl> <dbl> <lgl> <lgl> <lgl> <dbl> <dbl> <dbl>
1     1     2 FALSE FALSE FALSE     0    15     0
2     1     3 FALSE FALSE FALSE     0    24     0
```

```
3     1     4 FALSE FALSE FALSE     0     9     0
4     1     5 FALSE FALSE FALSE     0    18     0
5     1     6 FALSE FALSE FALSE     0    12     0
6     1     7 FALSE FALSE FALSE     0    33     0
```

in which, to recapitulate:

- `from` and `to` are vertex ids
- `p1` and `p2` are logical variables indicating whether researchers with ids `from` and `to` co-authored at least one publication in period 1 or period 2 respectively
- `test` variable is `TRUE` if researchers co-authored a publication in period 2, but not in period 1
- `aa`, `pa`, `cosi` columns contain scores for dyad `from-to` computed with measures, respectively: Adamic-Adar, Preferential Attachment, and Cosine based on L^+.

At this point we can use standard tools for evaluating classifier performance, such as the ROC curves, to analyze scores vis a vis the true labels in variable `test`. This is presented in the next section.

5.3. Results

To evaluate the predictive power of the three proximit measures we use ROC curves and package **ROCR** (Sing *et al.* 2005). First, we create a list of `prediction` objects with an element for each measure:

```
R> library(ROCR, warn.conflicts = FALSE)

Loading required package: gplots


Attaching package: 'gplots'

The following object is masked from 'package:stats':

    lowess

R> predlist <- lapply(
R+   c("aa", "pa", "cosi"),
R+   function(n) prediction(preds[[n]], preds$test)
R+ )
R> names(predlist) <- c("aa", "pa", "cosi")
```

Next, we are evaluating each measure by computing true positive and false positive rates. Plotting former agains the latter will create a ROC curve.

```
R> perflist <- lapply(predlist, performance, "tpr", "fpr")
```

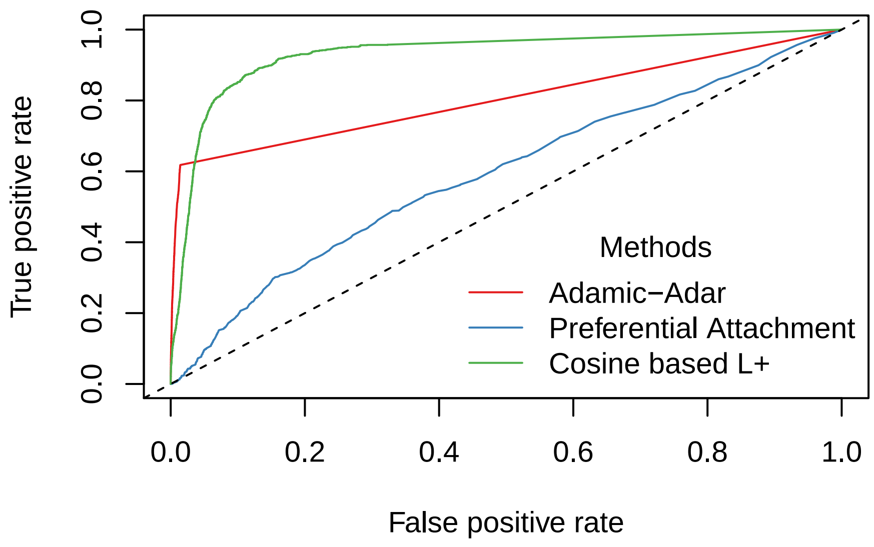

Figure 3: ROC curves for link-predictions based on three proximity indices: Adamic-Adar, Preferential Attachment, and Cosine based on L^+.

Finally, we plot all three curves in different color (using **RColorBrewer** by Neuwirth (2014)) on the same plot. The resulting curves are presented in Figure 3.

```
R> pal <- RColorBrewer::brewer.pal(3, "Set1")
R> for(i in seq(along=perflist)) {
R+   plot(
R+     perflist[[i]],
R+     col = pal[i],
R+     add = i != 1
R+   )
R+ }
R> abline(a=0, b=1, lty="dashed")
R> legend(
R+   "bottomright",
R+   title = "Methods",
R+   legend = c("Adamic-Adar", "Preferential Attachment", "Cosine based L+"),
R+   lty = 1,
R+   col = pal,
R+   bty = "n"
R+ )
```

We can extract Area Under Curve (AUC) values from the list of `performance` objects with:

```
R> vapply(
R+   predlist,
R+   function(p) performance(p, "auc")@y.values[[1]],
R+   numeric(1)
R+ )

       aa        pa      cosi
0.8031710 0.5914478 0.9316619
```

Clearly, the cosine based on L^+ performs best among the three measures compared.

Acknowledgements

Authors thank (Polish) National Science Centre for support through grant 2012/07/D/HS6/01971 for the project *Dynamics of Competition and Collaboration in Science: Individual Strategies, Collaboration Networks, and Organizational Hierarchies* (`http://recon.icm.edu.pl`).

References

Adamic L, Adar E (2001). "Friends and Neighbors on the Web." *Social Networks*, **25**, 211–230.

Backstrom L, Leskovec J (2011). "Supervised Random Walks: Predicting and Recommending Links in Social Networks." In *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 635–644. ACM.

Barabási AL (2016). *Network Science*. Cambridge University Press.

Barabási AL, Albert R (1999). "Emergence of Scaling in Random Networks." *Science*, **286**(5439), 509–512.

Bojanowski M, Chrol B (2018). *linkprediction: Link Prediction Methods*. R package version 1.0-0, URL `http://CRAN.R-project.org/package=linkprediction`.

Bojanowski M, Corten R (2014). "Measuring segregation in social networks." *Social Networks*, **39**, 14–32.

Brandes U, Robins G, McCranie A, Wasserman S (2013). "What Is Network Science?" *Network science*, **1**(1), 1–15.

Brin S, Page L (1998). "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *Computer Networks and ISDN Systems*, **30**(1–7), 107 – 117. Proceedings of the Seventh International World Wide Web Conference.

Butts CT (2008). "network: a Package for Managing Relational Data in R." *Journal of Statistical Software*, **24**(2). URL `http://www.jstatsoft.org/v24/i02/paper`.

Butts CT (2015). *network: Classes for Relational Data*. The Statnet Project (`http://statnet.org`). R package version 1.13.0.1, URL `http://CRAN.R-project.org/package=network`.

Chebotarev PY, Shamis EV (1997). "The Matrix-Forest Theorem and Measuring Relations in Small Social Groups." *Automation and Remote Control*, **58**(9), 1505–1514.

Clauset A, Moore C, Newman ME (2008). "Hierarchical Structure and the Prediction of Missing Links in Networks." *Nature*, **453**(7191), 98.

Csardi G, Nepusz T (2006). "The igraph software package for complex network research." *InterJournal*, **Complex Systems**, 1695. URL `http://igraph.org`.

Desmarais BA, Cranmer SJ (2013). "Forecasting the locational dynamics of transnational terrorism: A network analytic approach." *Security Informatics*, **2**(1), 8.

Ferligoj A, Kronegger L, Mali F, Snijders TA, Doreian P (2015). "Scientific collaboration dynamics in a national scientific system." *Scientometrics*, **104**(3), 985–1012.

Fouss F, Pirotte A, Renders JM, Saerens M (2007). "Random-Walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation." *IEEE Transactions on Knowledge and Data Engineering*, **19**(3), 355–369.

Goyal S (2012). *Connections: an Introduction to the Economics of Networks*. Princeton University Press.

Hanneke S, Fu W, Xing EP (2010). "Discrete Temporal Models of Social Networks." *Electronic Journal of Statistics*, **4**, 585–605.

Jaccard P (1912). "The Distribution of the Flora In The Alpine Zone." *New Phytologist*, **11**(2), 37–50.

Jackson MO (2010). *Social and Economic Networks*. Princeton University Press.

Katz L (1953). "A New Status Index Derived from Sociometric Analysis." *Psychometrika*, **18**(1), 39–43.

Klein DJ, Randić M (1993). "Resistance distance." *Journal of Mathematical Chemistry*, **12**(1), 81–95.

Krivitsky PN, Handcock MS (2014). "A separable model for dynamic networks." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **76**(1), 29–46.

Krivitsky PN, Handcock MS (2018). *tergm: Fit, Simulate and Diagnose Models for Network Evolution Based on Exponential-Family Random Graph Models*. The Statnet Project (`http://www.statnet.org`). R package version 3.5.2, URL `https://CRAN.R-project.org/package=tergm`.

Leicht EA, Holme P, Newman MEJ (2006). "Vertex similarity in networks." *Phys. Rev. E*, **73**(2), 026120.

Liben-Nowell D, Kleinberg J (2007). "The link-prediction problem for social networks." *Journal of the American Society for Information Science and Technology*, **58**(7), 1019–1031.

Lü L, Zhou T (2011). "Link prediction in complex networks: A survey." *Physica A*, **390**(6), 1150–1170.

McPherson M, Smith-Lovin L, Cook JM (2001). "Birds of a feather: Homophily in social networks." *Annual review of sociology*, **27**(1), 415–444.

Neuwirth E (2014). *RColorBrewer: ColorBrewer Palettes.* R package version 1.1-2, URL `https://CRAN.R-project.org/package=RColorBrewer`.

Newman M (2010). *Networks: An Introduction.* Oxford University Press.

Newman MEJ (2001). "Clustering and preferential attachment in growing networks." *Phys. Rev. E*, **64**(2), 025102.

PBN (2017). "Polska Bibliografia Naukowa (Polish Scholarly Bibliography)." URL `https://pbn.nauka.gov.pl/`.

Ravasz E, Somera AL, Mongru DA, Oltvai ZN, Barabási AL (2002). "Hierarchical Organization of Modularity in Metabolic Networks." *Science*, **297**(5586), 1551–1555.

Ripley RM, Snijders TAB, Boda Z, Vörös A, Preciado P (2018). "Manual for Siena version 4.0." *Technical report*, Oxford: University of Oxford, Department of Statistics; Nuffield College. R package version 1.2-12. https://www.cran.r-project.org/web/packages/RSiena/.

Sing T, Sander O, Beerenwinkel N, Lengauer T (2005). "ROCR: visualizing classifier performance in R." *Bioinformatics*, **21**(20), 7881. URL `http://rocr.bioinf.mpi-sb.mpg.de`.

Snijders TAB (1996). "Stochastic actor-oriented models for network change." *Journal of mathematical sociology*, **21**(1-2), 149–172.

Sørensen T (1948). "A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons." *Biologiske Skrifter*, **5**, 1–34.

Van de Bunt GG, Van Duijn MA, Snijders TA (1999). "Friendship networks through time: An actor-oriented dynamic statistical network model." *Computational & Mathematical Organization Theory*, **5**(2), 167–192.

Wasserman S, Faust K (1994). *Social network analysis: Methods and applications*, volume 8. Cambridge university press.

Wickham H, François R, Henry L, Müller K (2018). *dplyr: A Grammar of Data Manipulation.* R package version 0.7.7, URL `https://CRAN.R-project.org/package=dplyr`.

Wickham H, Henry L (2018). *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions.* R package version 0.8.1, URL `https://CRAN.R-project.org/package=tidyr`.

Zhou T, Lü L, Zhang YC (2009). "Predicting missing links via local information." *The European Physical Journal B*, **71**(4), 623–630.

Affiliation:

Michał Bojanowski
Kozminski University
Jagiellońska 57/59, 03-301 Warsaw, Poland
E-mail: `mbojanowski@kozminski.edu.pl`

Bartosz Chroł
ICM, University of Warsaw
Tyniecka 15/17, 02-630 Warsaw, Poland
E-mail: `bartek.chrol@gmail.com`