Internal link prediction: a new approach for predicting links in bipartite graphs

Oussama Allali, Clémence Magnien and Matthieu Latapy

LIP6 – CNRS and Université Pierre et Marie Curie (UPMC – Paris 6)
4 place Jussieu
75252 Paris cedex 05– France
firstname.lastname@lip6.fr

Abstract

Many real-world complex networks, like actor-movie or file-provider relations, have a bipartite nature and evolve over time. Predicting links that will appear in them is one of the main approach to understand their dynamics. Only few works address the bipartite case, though, despite its high practical interest and the specific challenges it raises. We define in this paper the notion of internal links in bipartite graphs and propose a link prediction method based on them. We thoroughly describe the method and its variations, and experimentally compare it to a basic collaborative filtering approach. We present results obtained for a typical practical case. We reach the conclusion that our method performs very well, and we study in details how its parameters may influence obtained results.

1 Introduction

Many real-world complex networks have a natural bipartite structure and may therefore be modeled as bipartite graphs, *i.e.* two sets of nodes with links only between nodes in different sets. Typical examples include actor-movie networks [40] where actors are linked to the movies they played in; authoring networks [30] where authors are linked to the papers they co-signed; peer-to-peer exchange graphs [17] where peers are linked to the files they provided/searched; and on-line shopping networks where clients are linked to the products they bought [28].

Studying such networks has received recently much attention, see [25] for a survey. The main approach consists in transforming bipartite graphs into classical (non-bipartite) graphs through a process called *projection*[1], but this induces an important loss of information. Studying them directly as bipartite graphs therefore is very appealing, but there is a lack of established methods for doing so.

In addition most of these networks are dynamic: they evolve over time, with node and link additions and removals. Studying such dynamics is extremely important for our understanding of these objects, but here again very limited knowledge and basic methods is available, even for classical (non-bipartite) dynamic graphs.

One of the main approaches developed for studying graph dynamics is *link prediction* [27, 18, 6], which consists in predicting the links that will probably appear in the future, given a snapshot of the considered graph at a given time.

[1] One studies for instance co-starring networks, where two actors are linked if they played together in a movie, or co-authoring networks where two authors are linked if they signed a paper together.

We address here the problem of link prediction in dynamic bipartite graphs. To do so, we introduce a special kind of links in bipartite graphs, which we call *internal links*. We then propose an approach based on these links and compare it to a basic classical approach. We study the performance of our method on real-world datasets using wide ranges of parameters and present the results for a file-provider graph. This shows that this method reaches very good performances and that internal links play a key role in the dynamics of real-world bipartite graphs.

The paper is organized as follows. We review related work in Section 2 and present the bipartite framework, including the notion of internal links, in Section 3. We then formally state the considered problem and its assessment in Section 4. We present in Section 5 our prediction method and a basic method which we consider for comparison purpose. We finally present our experimental setup in Section 6 and the results of experiments in Section 7. We discuss our conclusions and perspectives in Section 8.

2 Related work

Link prediction is a key research problem in network dynamic analysis. Several works study this problem on classical (non-bipartite) graphs. Most of them are based on measures of similarity between nodes. For instance, in [27] the authors examine several topological measures (such as Jaccard coefficient, Adamic/Adar coefficient, SimRank, etc.) based on node neighborhoods and the set of all paths between nodes. They use these measures for ranking possible future co-authors collaborations. In [20] the author proposes to use another topological measure called generalized clustering coefficient. In [18, 31] the authors add several non-topological measures based on node attributes (such as keyword match, number of papers, geographic proximity, KL-divergence of two nodes' topic distribution, etc.) and they use a supervised learning algorithm to perform link prediction. In a similar way, the authors of [7] predict co-authoring of publications by using topological measures computed in the co-authoring graph and indirect topological measures computed using the co-author graph (where two papers are linked if they are signed by a same author). The authors of [39] add another measure (local probabilistic model) to estimate the co-occurrence probability of two nodes, and in [37] the authors extend this by incorporating available temporal information. Finally, the authors of [11] use a hierarchical decomposition of a social network and use it for predicting missing links. They generate a set of hierarchical random graphs, and they compute average probability of connection between two nodes within these hierarchical random graphs.

Works presented above deal with classical (non-bipartite) graphs, and are not directly applicable to bipartite graphs. In [21], the authors adapt some topological measures used in classical graphs for predicting links in bipartite graphs. For each possible link (u, v), the authors compare the neighbors of u in the bipartite graph and neighbors of neighbors of v. They also study the set of all paths between nodes, in the bipartite graph. Going further, the authors of [6] consider two transformations of the bipartite graph into a classical one, then for predicting link (u, v) they consider the classical graph containing u, and they study the topological measures between u and the neighbors of v in the bipartite graph, and conversely.

The authors of [13] study two prediction problems in bipartite graphs in which links appear and disappear over time: *predicting new links* aims at predicting links that will appear and were never observed before; *predicting all links* aims at predicting the links that will exist in the considered period, including links that were previously observed. They use matrix- and tensor based methods to do so. In [24, 23], the authors study link prediction in growing (bipartite) graphs. They argue that, in practice, the growth changes the eigenvalues but leaves

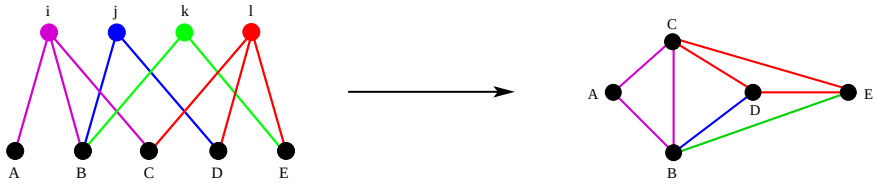


Figure 1: An example of bipartite graph G (left), and its $\perp$-projection $G_\perp$(right).

the eigenvectors largely unchanged. Studying the eigenvalues evolution allows to propose a growth model for a given network, and use it for link prediction.

Another research problem is closely related to link prediction in bipartite graphs: the recommendation problem [34]. Recommendation systems are used to suggest items to users, such as products to customers for instance. Notice however that the two problems are quite different: recommendation aims typically at finding a small number of products of interest for each customer; prediction aims at finding links that will appear in the future. Predicting that a given node will have a huge number of new links while many others will not have any is of little interest regarding recommendation but may be a great success regarding prediction.

Various approaches have been developed for recommendation [5, 3, 22], with collaborative filtering being the most successful and widely used approach [22]. Two main approaches of collaborative filtering have been proposed, both based on the idea that similar users will purchase similar items and that users will purchase items similar to the ones they already purchased. The first approach consists in predicting the rating of a given user for a given item [33]. It relies on known rating data (e.g. explicit users opinions for items, rated on a scale of 1 to 5). The second approach consists in ranking the most relevant items for a given user in order of decreasing interest, and then in recommending the top N items to this user [12, 28]. This approach does not require explicit ratings but only the information of which users adopted which items, and is the most similar to link prediction. We will use such an approach in this paper for the purpose of comparison with our method, see Section 5.2.

3 The bipartite framework

We present here bipartite graphs and their transformations into (weighted) classical graphs, called projection. We also introduce a new class of links in bipartite graphs, which we call *internal links*. These links are at the core of our work.

3.1 Bipartite graphs, projections, and internal links

A bipartite graph $G = (\perp, \top, E)$ is defined by a set $\perp$ of bottom nodes, a set $\top$ of top nodes and a set $E \subseteq \perp \times \top$ of links. The key point is that links exist only between a node in $\perp$ and one in $\top$. We denote by $N(u) = \{v \in (\perp \cup \top),\ (u, v) \in E\}$ the neighborhood of a node u in G. If $u \in \perp$ then $N(u) \subseteq \top$, and conversely. More generally, given any set of nodes $S \subseteq (\perp \cup \top)$, we denote by $N(S)$ its neighborhood: $N(S) = \bigcup_{u \in S} N(u)$.

The $\perp$-projection of G is the graph $G_\perp = (\perp, E_\perp)$ in which $(u, v) \in E_\perp$ if u and v have at least one neighbor in common in G: $N(u) \cap N(v) \neq \emptyset$. In other words, there is a link between u and v in $G_\perp$ if u and v are linked to a same top node in G. As a consequence, each top node induces in $G_\perp$ a clique between its neighbors in G. See Figure 1 for an example. We denote by

$N_\perp(u)$ the neighborhood of a node u in $G_\perp$: $N_\perp(u) = \{v \in \perp,\ (u,v) \in E_\perp\} = N(N(u))$. The $\top$-projection of G, denoted by $G_\top$, is defined dually.

We now introduce a special class of links, called *internal links*, which play a key role in the whole paper.

Definition 1 (internal links) *Let us consider a bipartite graph $G = (\perp, \top, E)$ and the bipartite graph $G' = (\perp, \top, E \cup \{(u,v)\})$ obtained by adding the link $(u,v) \in \perp \times \top$ to G, with $(u,v) \notin E$. The link (u,v) is* internal *if $G_\perp = {G'}_\perp$.*

In other words, an internal link in a bipartite graph G is a pair of nodes (u,v) such that adding the link (u,v) to G does not change its $\perp$-projection. In Figure 1 for instance, (B,l) is an internal link. Indeed, all neighbors of l in G, namely $N(l) = \{C,D,E\}$, are already linked to B in $G_\perp$: the pairs of bottom nodes (B,C), (B,D) and (B,E) already have a neighbor in common in G, respectively, i, j and k. Adding link (B,l) to G increases their number of common neighbors to 2 and thus does not change $\perp$-projection.

As this example indicates, internal links have a characterization in terms of neighborhood which we give now.

Lemma 1 *Given a bipartite graph $G = (\perp, \top, E)$, a pair of nodes (u,v) in $(\perp \times \top) \setminus E$ is an internal link for G if and only if $N(v) \subseteq N(N(u))$.*

Proof : Let us consider a pair of nodes (u,v) in $(\perp \times \top) \setminus E$ and let $G' = (\perp, \top, E' = E \cup \{(u,v)\})$ be the bipartite graph obtained by adding the link (u,v) to G. Then, by definition, ${E'}_\perp = E_\perp \cup \{(u,x),\ x \in N(v)\}$.

Suppose now that (u,v) is an internal link, *i.e.* $E_\perp = {E'}_\perp$. Then all links (u,x) in the expression above already belong to $E_\perp$. Therefore, for each $x \in N(v)$, there exists y in $\top$, such that $y \in N(u) \cap N(x)$. By symmetry, $x \in N(y)$ and $y \in N(u)$ therefore, $x \in N(N(u))$ and so $N(v) \subseteq N(N(u))$.

Suppose now that $N(v) \subseteq N(N(u))$. Then for each node x in $N(v) \subseteq N(N(u))$, there exists y in $\top$, such that x is a neighbor of y and y is a neighbor of u. Thus, by definition of the projection, $(u,x) \in E_\perp$, and so $E_\perp = E'_\perp$ and the link (u,v) is internal.

□

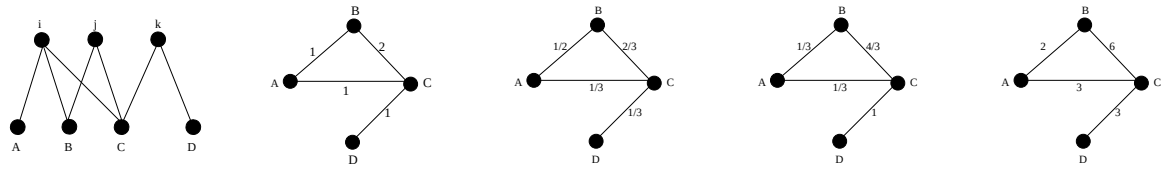
We finally introduce the notion of induced links, which will be useful in the following.

Definition 2 (induced links) *Given a bipartite graph $G = (\perp, \top, E)$, the set of links induced by any pair of nodes (u,v) in $(\perp \times \top)$ is: $\perp(u,v) = \{u\} \times N(v) = \{(u,w), w \in N(v)\}$.*

In Figure 1, for instance, $\perp(A,j) = \{A\} \times N(j) = \{(A,B),(A,D)\}$. Notice that $E_\perp = \bigcup_{(u,v)\in E} \perp(u,v)$: the links of the $\perp$-projection of G are the links induced by all the links of G. By definition, a pair of nodes $(u,v) \in (\perp \times \top) \setminus E$ is an internal link if and only if all the links it induces are already in $G_\perp$. In Figure 1, for instance, $\perp(B,l) = \{(B,C),(B,D),(B,E)\} \subseteq E_\perp$ and therefore (B,l) is an internal link.

3.2 Weighted projections

As explained for instance in [25], $G_\perp$ contains much less information than G. In particular, the fact that u and v are linked in $G_\perp$ means that they have *at least* one neighbor in common in G but says nothing on their *number* of common neighbors. Several approaches are used for weighting the links of the $\perp$-projection in order to capture such information. We present the main ones in this section (examples are displayed in Figure 2).



G $(G_\perp, \sigma)$ $(G_\perp, \gamma)$ $(G_\perp, \delta)$ $(G_\perp, \rho)$

Figure 2: A bipartite graph G (left) and its $\perp$-projection with the different weight functions defined in Section 3.2.

First, the weight of link (u, v) may be defined as the number of (top) neighbors that u and v have in common in the bipartite graph, called *sum*:

$$\sigma(u, v) = |N(u) \cap N(v)|.$$

The σ weight function has been used for instance to estimate the probability of collaboration between authors [29].

Notice that if u and v both have many neighbors, then $\sigma(u, v)$ will naturally tend to be high. Conversely, if u and v have only few neighbors but these neighbors are the same, then $\sigma(u, v)$ is low, which does not reflect the fact that u and v are very similar. To capture this, one may use the *Jaccard* coefficient:

$$\gamma(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}.$$

This quantity has been used for instance in the context of peer-to-peer exchange analysis to capture similarity between peers [26].

The value of $\gamma(u, v)$ may however be strongly biased if one of the two nodes has many neighbors and the other one only few: the value would then be very low, even if all neighbors of one node are neighbors of the other. To avoid this, variants of the *Jaccard* coefficient, *overlap* and *cosine*, have been proposed in the literature [36, 35]:

$$\gamma^{ove}(u, v) = \frac{|N(u) \cap N(v)|}{\min(|N(u)|, |N(v)|)} \qquad \gamma^{cos}(u, v) = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)| \times |N(v)|}}.$$

From this point of view, though, nodes play an unbalanced role: a $\top$-node x has an influence on the similarity between $\frac{|N(x)| \times (|N(x)|-1)}{2}$ pairs of $\perp$-nodes. When $N(x)$ is large, this is huge; on the contrary, if a $\top$-node only has two neighbors then it probably indicates a significant similarity between them. To capture this, one may consider that each $\top$-node *votes* for the similarity between its neighbors and that the sum of its votes is only one (it has only one voice to distribute). This leads to the *delta* function:

$$\delta(u, v) = \sum_{x \in N(u) \cap N(v)} \frac{2}{|N(x)| \times (|N(x)| - 1)}.$$

In Figure 2, for instance, nodes i and j vote respectively $\frac{1}{3}$ and 1 for link (B, C), so $\delta(B, C) = \frac{1}{3} + 1$. A similar quantity has been used in [1] to capture the similarity between two home pages as a function of the features they share.

All weight functions above intuitively capture similarity between nodes. One may also use weight functions to capture other features, like the activity of nodes in the network. This leads

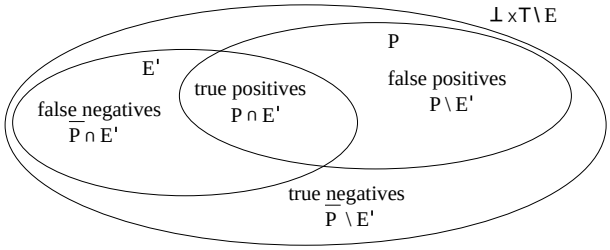


Figure 3: A prediction method divides the set of possible links $\perp \times \top \setminus E$ into four categories: true positives, $P \cap E'$; true negatives, $\overline{P} \setminus E'$; false positives, $P \setminus E'$; and false negatives, $\overline{P} \cap E'$.

for instance to compute the product of the number of neighbors of u and v in the bipartite graph [4, 29], called *attachment*:

$$\rho(u, v) = |N(u)| \cdot |N(v)|,$$

which reflects the expectation that u and v may have neighbors in common: if links were placed at random then the probability that (u, v) is a link would be proportional to $\rho(u, v)$.

All weighting functions presented above are natural and capture relevant informations about a bipartite graph. Each has its own strengths and weaknesses, and up to our knowledge there has been only limited comparison between them until now. By comparing their performance in the context of link prediction below, we expect to give some insight on their respective relevance in this context, see in particular Section 7.3.

4 The bipartite link prediction problem

Let us consider a dynamic bipartite graph defined by a set of n timestamped links $D = \{(t_i, u_i, v_i), i = 1...n\}$. Let $G = (\perp, \top, E)$ be the graph observed from a given instant a to another instant $b > a$: $\perp = \{u, \exists (t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$, $\top = \{v, \exists (t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$ and $E = \{(u, v), \exists (t, u, v) \in D \text{ s.t. } a \leqslant t < b\}$. We call G the *reference graph* and $[a, b[$ the *reference period*.

Now let us consider an instant $c > b$. This induces a set E' of links added to G during the period $[b, c[$, which we call the *prediction period*: $E' = \{(u, v), \exists (t, u, v) \in D \text{ s.t. } b \leqslant t < c\} \cap (\perp \times \top \setminus E)$. Notice that we consider only the links between nodes of G (we ignore new nodes appearing in the period $[b, c[$) which are not present in G (we consider links in $\perp \times \top \setminus E$ only).

In this framework, the goal of a link prediction method is to find a set P of *predicted links* which contains many of the links in E' but only few which are not in E'. Notice that in the extreme case where one predicts *all* possible links, *i.e.* $P = \perp \times \top \setminus E$, then one succeeds in predicting all links of E' but also predicts many links which are not in E'. Conversely, predicting no link at all, *i.e.* $P = \emptyset$, trivially does not predicting links not in E' but fails in predicting any link in E'.

Evaluating the performances of a prediction method therefore consists in evaluating its success in reaching a tradeoff regarding these two objectives, which is non-trivial. We present below a classical method to do so [36, 10], which we use in this paper.

Let us denote by $\overline{P}$ the set of links that the method predicts and will not appear: $\overline{P} = (\perp \times \top \setminus E) \setminus P$. Figure 3 illustrates the four possible cases which may occur during link prediction: the set $P \cap E'$ of *true positives* is the set of appearing links that the method successfully predicts; the set $\overline{P} \setminus E'$ of *true negatives* is the set of unpredicted links which indeed do not appear;

conversely, the *false positives* are the links in $P \setminus E'$, *i.e.* the links which we predicted but do not appear, and the *false negatives* are the links in $\overline{P} \cap E'$.

The aim of a link prediction method is to maximize the number of true positives and negatives while minimizing the number of false positives and negatives. This is captured by two quantities, called *precision* and *recall*.

The *precision* is the fraction of true positives among the predicted links, *i.e.* $\frac{|P \cap E'|}{|P|}$. In other words, it is the probability that the method is right when it predicts that a given link will appear, and therefore is a measure of correctness.

The *recall* is the fraction of true positives among the appearing links, *i.e.* $\frac{|P \cap E'|}{|E'|}$. In other words, it is the probability that an appearing link will indeed be predicted by the method, and so is a measure of completeness.

As explained above, there is a tradeoff between precision and recall, as, in general, improving one degrades the other and conversely. In order to capture this in a single value, which often is more convenient, one generally considers the *F-measure*, $\frac{2 \times |P \cap E'|}{|P| + |E'|}$, which is the harmonic mean of precision and recall [38]. The goal of a prediction method then is to maximize the F-measure.

5 Bipartite prediction methods

In this section, we introduce our link prediction method for bipartite graphs, which we call *internal link prediction*. We also present a typical collaborative filtering method which we use for comparison in the next sections.

5.1 Internal link prediction

The key feature of our prediction method is that it focuses on internal links: it predicts internal links only. The intuition behind this is that two bottom nodes which already have a common neighbor in G (*i.e.* they are linked in $G_\perp$) will probably acquire more in the future. Instead, if two nodes have no common neighbor in G, then they will probably still have none in the future. The links that can be added to G which fit both criteria are precisely internal links. We will see in Section 6.2 that a large fraction of the links which appear are indeed internal.

Going further, two bottom nodes with many common neighbors in G will probably have more in the future. More generally, all the weight functions presented in Section 3.2 are measures (from different points of view) of our expectation that two nodes having neighbors in common probably will have more in the future. Therefore, we expect that the links that will appear are the internal links inducing $\perp$-links with high weights.

This leads to the following prediction method, which we call *internal links prediction*. Let us consider a weight function ω like the ones described in Section 3.2, and a given weight threshold τ. We denote by $E_{\perp\tau} = \{(u, w) \in E_\perp,\ \omega(u, w) \geq \tau\}$ the set of links in the projection that have a weight larger than or equal to τ. We then predict all the internal links which induce at least one link in $E_{\perp\tau}$.

Figure 4 shows an example of internal link prediction using the *Jaccard* weight function, γ. The set of internal links of G is $\{(B, l), (C, k), (D, k), (E, j)\}$; let us focus on the internal link (B, l). It induces (B, C), (B, D), and (B, E). Given a threshold τ we predict (B, l) if one of these links has weight at least τ. For instance:

- if $\tau = \frac{1}{4}$, all links in the projection have a weight larger than or equal to τ, and so we predict all possible internal links in the bipartite graph, including (B, l);
- if $\tau = \frac{1}{3}$, only 5 links in the projection have weight larger than or equal to τ, including (B, C), which is induced by (B, l); we therefore predict (B, l);

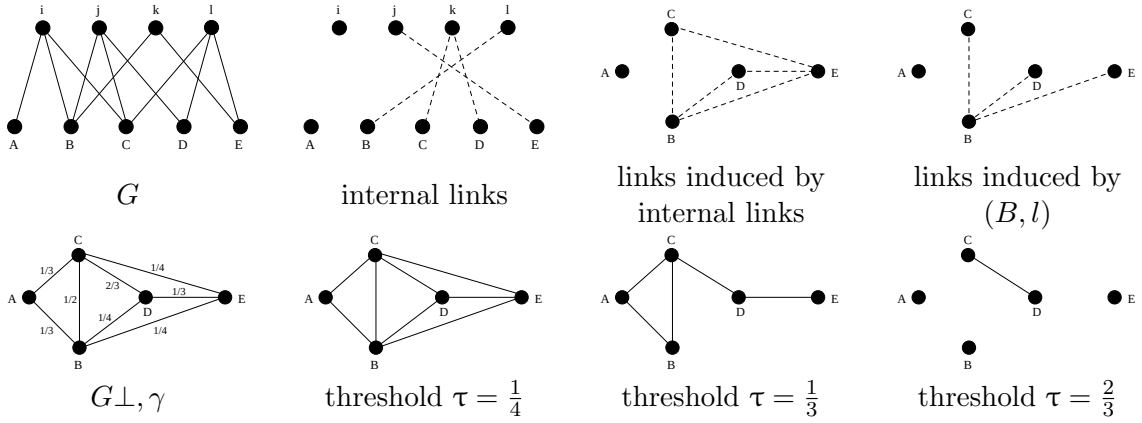


Figure 4: **Example of internal link prediction**. First row (left to right): a bipartite graph G, the internal links of G, the $\perp$-links they induce, and the links of $G_{\perp}$ induced by the internal link (B, l). Second row (left to right): the *Jaccard* weighted $\perp$-projection of G ($G\perp, \gamma$), and the links $E_{\perp \frac{1}{4}}$, $E_{\perp \frac{1}{3}}$, $E_{\perp \frac{2}{3}}$ obtained using thresholds τ respectively equal to $\frac{1}{4}$, to $\frac{1}{3}$ and to $\frac{2}{3}$.

- if $\tau = \frac{2}{3}$, only one link has a weight larger than or equal to τ, and it is not a link induced by (B, l); therefore we do not predict (B, l).

Algorithm 1 provides the details of the method useful for implementation, and Theorem 1 shows its complexity.

Algorithm 1: Internal link prediction

Input: bipartite graph $G = (\perp, \top, E)$, weight function ω, threshold τ
Output: set P of predicted links

```
P ← ∅
for u ∈ ⊥ do
    Nbot ← ∅
    for v ∈ N(u) do
        for w ∈ N(v)\{u} do
            Nbot ← Nbot ∪ {w}
            compute ω(u, w)
    I_u ← ∅
    d ← 0, 0, ..., 0
    m ← ∅
    for w ∈ Nbot do
        for v ∈ N(w) do
            if ω(u, w) ⩾ τ then
                m ← m ∪ {v}
            d[v] ← d[v] + 1
    for v ∈ m do
        if |N(v)| = d[v] then
            I_u ← I_u ∪ {v}
    P ← P ∪ {(u, v), v ∈ I_u}
```

Theorem 1 *Algorithm 1 performs internal link prediction on a bipartite graph $G = (\bot, \top, E)$ in time $\mathcal{O}(\Delta_\bot |E|)$, where $\Delta_\bot = \max_{u \in \bot} |N_\bot(u)|$ is the largest degree in $G_\bot$, and space $\mathcal{O}(|\top| + |\bot|)$ in addition to the space needed for storing G.*

Proof : We first show the termination and correctness of our algorithm. The algorithm consists of imbrications of `for` loops over finite and static sets, so it necessarily terminates.

To show its correctness, we must show that it predicts all the internal links that induce links in $E_{\bot\tau}$, *i.e.* links with weight larger than or equal to τ, and only those links.

The algorithm consists in a loop over all nodes u in $\bot$, with three main parts, each consisting in a `for` loop.

The first part computes $N_\bot(u)$, denoted by *Nbot* in the algorithm to emphasize that it is not precomputed nor stored, and the weights of the corresponding links. The nodes w added to *Nbot* are exactly those in $N(N(u))$, which by definition is exactly $N_\bot(u)$. Therefore $Nbot = N_\bot(u)$ at the end of the first part of the loop. We do not detail the weight computation here, because it depends on the weight function considered. We assume that it can be computed with the same time complexity overhead as $N_\bot$, which is true for all the weight functions presented in Section 3.2.

The second part does two things. First it stores in a set m the nodes v such that (u, v) induces a link $(u, w) \in E_{\bot\tau}$ (notice that (u, v) is not necessarily an internal link). Indeed, a node v is added to m as soon as it has a neighbor $w \in \bot$ such that $(u, w) \in E_{\bot\tau}$.

This loop also stores in $d[v]$ the number of neighbors of v which are neighbors of u in $G_\bot$. At the end of the second part of the loop, the nodes v for which $d[v] = |N(v)|$ are exactly the nodes v such that (u, v) is an internal link. Indeed, $d[v]$ is incremented for each $w \in N(v)$ such that $w \in N_\bot(u)$. Therefore $d[v] = |N(v)|$ if and only if $N(v) \subseteq N_\bot(u) = N(N(v))$, *i.e.* if and only if (u, v) is an internal link, from Lemma 1.

The third part of the loop then computes the intersection between the nodes v which correspond to internal links (those for which $d[v] = |N(v)|$) and the nodes corresponding to links which induce at least one link $(u, w) \in E_{\bot\tau}$ (the nodes in m).

Before entering in the details of the complexity analysis, let us discuss how sets may be efficiently managed in our algorithm. A set s of nodes in $\top$ (resp. $\bot$) may be represented by an array indexed by nodes in $\top$ (or $\bot$), such that $s[v] = 1$ if and only if $v \in s$, together with an array i_s containing the indexes of nodes in s, and an integer n_s representing the number of nodes in s. Therefore listing all nodes in s simply consists in listing the n_s first values of i_s. Adding a node v to s is performed in two steps: if $s[v] = 1$, do nothing; otherwise, set $s[v]$ to 1, increment n_s, and set $i_s[n_s]$ to v. To reinitialize s to $\emptyset$, iterate set $s[i_s[k]] \leftarrow 0$ for all $k \leq n_s$, then set $n_s = 0$. Finally, adding an element to a set requires constant time, and setting a set to $\emptyset$ requires as many operations as $|s|$, which is necessarily bounded by the time needed to populate the set, and therefore does not create any time complexity overhead. In our algorithm, the array d may be managed in a similar way: an additional array stores the indexes of nodes for which $d[v] \neq 0$, which allows to reset d to 0 without iterating over all nodes in $\top$.

This leads to the space complexity of our algorithm. With the encoding above, a set of nodes in $\top$ (resp. $\bot$) requires $\Theta(|\top|)$ (resp. $\Theta(|\bot|)$) space. Therefore, although the number of links in $G_\bot$ is huge in general compared to $|E|$, our algorithm only requires $\Theta(|\top| + |\bot|)$ space in addition to the space needed for storing its input G and its output P (which we may choose not to store).

Let us now study the time complexity of the algorithm. The main loop runs over all nodes in $\bot$ and performs three sets of operations. We will study the complexity of these parts independently.

The first part of the loops performs $\Theta(|N(v)|)$ operations for each node $v \in N(u)$. Therefore the total number of operations for the first part of the main loop is of the order:

$$\Theta\left(\sum_{u\in\perp}\sum_{v\in N(u)}|N(v)|\right) = \Theta\left(\sum_{v\in\top}|N(v)||N(v)|\right).$$

We have that $|N(v)| \leq \delta_\top$, where $\delta_\top = \max_{v\in\top}|N(v)|$. Therefore the above quantity can be bounded:

$$\Theta\left(\sum_{v\in\top}|N(v)||N(v)|\right) \subseteq \mathcal{O}\left(\delta_\top\sum_{v\in\top}|N(v)|\right) = \mathcal{O}\left(\delta_\top|E|\right).$$

The second part of the loop performs $\Theta(|N(w)|)$ operations for each node $w \in N_\perp(u)$. The total number of operations in this part of the loop is therefore performed in time:

$$\Theta\left(\sum_{u\in\perp}\sum_{w\in N_\perp(u)}|N(w)|\right) = \Theta\left(\sum_{w\in\perp}|N_\perp(w)||N(w)|\right).$$

This expression can be bounded in two ways, by considering that $|N_\perp(w)| \leq \Delta_\perp$ (where $\Delta_\perp = \max_{u\in\perp}|N_\perp(u)|$ is the largest degree in $G_\perp$) or that $|N(w)| \leq \delta_\perp$ (where $\delta_\perp = \max_{v\in\perp}|N(v)|$). In the first case we obtain

$$\Theta\left(\sum_{w\in\perp}|N_\perp(w)||N(w)|\right) \subseteq \mathcal{O}\left(\Delta_\perp\sum_{w\in\perp}|N(w)|\right) = \mathcal{O}\left(\Delta_\perp|E|\right).$$

In the second case we obtain

$$\Theta\left(\sum_{w\in\perp}|N_\perp(w)||N(w)|\right) \subseteq \mathcal{O}\left(\delta_\perp\sum_{w\in\perp}|N_\perp(w)|\right) = \mathcal{O}\left(\delta_\perp|E_\perp|\right).$$

Finally, the third part of the loop iterates over all nodes in m. Since m was computed in the second part of the loop, the number of operations in the third part of the loop is bounded by the one for the second part of the loop, therefore we do not need to evaluate it further.

The overall complexity of the algorithm is therefore in the order of

$$\mathcal{O}\left(\delta_\top|E| + \min(\Delta_\perp|E|, \delta_\perp|E_\perp|)\right) \subseteq \mathcal{O}\left((\delta_\top + \Delta_\perp)|E|\right) \subseteq \mathcal{O}\left(\Delta_\perp|E|\right),$$

because $\delta_\top \in \mathcal{O}(\Delta_\perp)$ since each node in $\top$ with degree d induces a clique of d nodes, each of them having a degree at least $d-1$ in $G_\perp$.

□

5.2 Collaborative filtering prediction

As explained in Section 2, typical collaborative filtering approaches consist in predicting that $\perp$-nodes tend to create links to the $\top$-neighbors of $\perp$-nodes which are similar to themselves (clients will buy products that similar clients already bought). More precisely, such methods first select for each $\perp$-node u the set of k $\perp$-nodes which are the most similar to u and then the N $\top$-nodes the most strongly linked to these nodes, for given parameters k and N. Here, natural notions of similarity between $\perp$-nodes are provided by the weighted $\perp$-projection.

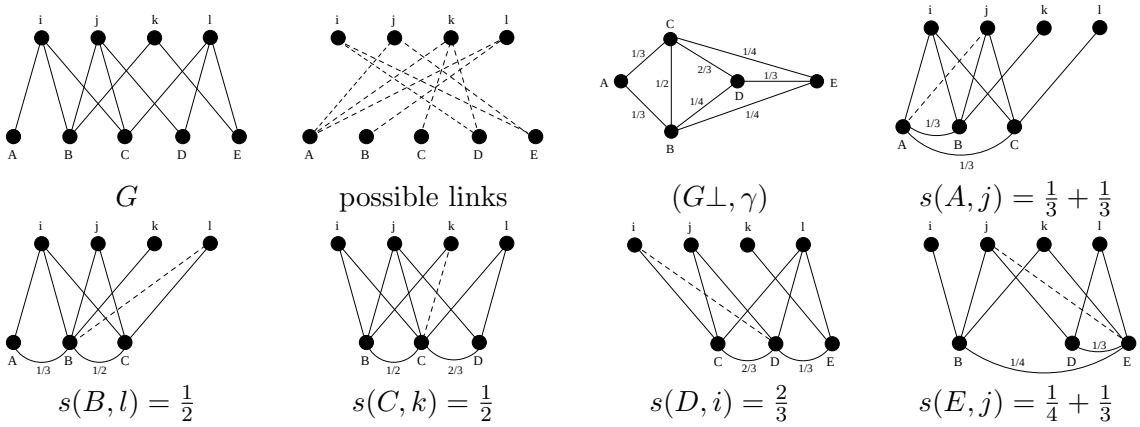


Figure 5: **Example of collaborative filtering prediction**. First row (left to right): an example of bipartite graph G, the set of all possible links that may appear, the *Jaccard* weighted $\perp$-projection of G $(G\perp, \gamma)$, and the result of collaborative filtering prediction for node A. Second row (left to right): the result of collaborative filtering prediction for B, C, D and E. In this example, $k = 2$ (we consider 2 most similar neighbors) and $N = 1$ (we predict 1 link with highest score).

We will therefore use the following collaborative filtering method [8]: given a weight function ω, we consider for each $\perp$-node u the set $U_k \subseteq N_\perp(u)$ of its k neighbors with largest weight. Then for each $v \in N(U_k) \backslash N(u)$, we compute the score $s(u, v)$ of the link (u, v) as the sum of the weights $\omega(u, w)$, for each $w \in N(v) \cap U_k$:

$$s(u, v) = \sum_{w \in U_k \cap N(v)} \omega(u, w).$$

There are other possible ways to compute the score [14, 41, 9, 15, 19], however in this paper we restrict ourselves to the formula above which is typical. Finally, the collaborative filtering method predicts for each node the N links with highest scores.

Figure 5 presents an example of collaborative filtering using the *Jaccard* weight function γ, $k = 2$ and $N = 1$. For instance, node E has three neighbors in the $\perp$-projection: $N_\perp(E) = \{B, C, D\}$. The method then considers the $k = 2$ most similar ones, *i.e.* the ones linked to E with the largest weight in $G_\perp$. As (E, B) and (E, C) have the same weight, the method chooses at random between B and C. Suppose that $U_2 = \{B, D\}$. The method then computes the score of links (E, v) for all $v \in (N(B) \cup N(D)) \backslash N(E)$, leading to $s(E, j) = \gamma(E, B) + \gamma(E, D) = \frac{7}{12}$, and $s(E, i) = \gamma(E, B) = \frac{1}{4}$. Finally, as $N = 1$, it predicts the link with highest score, (E, j).

Algorithm 2: Collaborative filtering

Input: bipartite graph $G(\bot, \top, E)$, weight function ω, k, N
Output: set P of predicted links

$P = \emptyset$
for $u \in \bot$ **do**
  $Nbot \leftarrow \emptyset$
  for $v \in N(u)$ **do**
    for $w \in N(v) \backslash \{u\}$ **do**
      $Nbot \leftarrow Nbot \cup \{w\}$
      compute $\omega(u, w)$
  $U \leftarrow \text{sort}(Nbot, \omega)$
  $U_k \leftarrow \text{head}(U, k)$
  $s \leftarrow 0, 0, \ldots, 0$
  for $w \in U_k$ **do**
    for $v \in N(w) \backslash N(u)$ **do**
      $s[v] \leftarrow s[v] + \omega(u, w)$
  $P_u \leftarrow \text{sort}(s)$
  $P \leftarrow P \cup \{(u, v), v \in \text{head}(P_u, N)\}$

We present details in Algorithm 2. The first part of the loop computes $N_\bot(u)$. The corresponding time complexity is therefore in the order of $\mathcal{O}\left(\delta_\top |E|\right) \subseteq \mathcal{O}(\Delta_\bot |E|)$, as detailed in the proof of Theorem 1 (notations are defined there). The complexity of the second part of the loops depends on parameters k and N. The `sort` and `head` instructions are used to compute the k and N largest values in $N_\bot$ (according to weight function ω) and s, respectively. This can be done in constant time if k and N are constant, but we kept this notation to make the algorithm easier to read.

The loop iterating over all nodes $w \in U_k$ performs $|N(w)|$ operations at each step. This loop is repeated for all nodes $u \in \bot$. The total number of operations performed is bounded by $\mathcal{O}(\sum_{u \in \bot} \sum_{w \in N_\bot(u)} |N(w)|) \subseteq \mathcal{O}(\Delta_\bot |E|)$.

The total time complexity is therefore in the order of $\mathcal{O}(\Delta_\bot |E|)$, as Algorithm 1. The space complexity, besides the space needed to store G, is of the order of $|\top| + |\bot|$, this space being needed for storing $Nbot, U, U_k$, and s. This is the same as Algorithm 1.

6 Experimental setup

Evaluating our method in practice requires the availability of large scale bipartite data *with their dynamics*. One natural source for such data might be benchmarks for recommendation systems. However, such datasets often do not contain temporal information or have been filtered to fit recommendation needs (for instance, nodes with a small degree have been removed, as well as large degree ones). This makes them unusable in our context.

We finally conducted experiments on various datasets, in particular user-tag graphs from delicious[2] [16] as well as user-group and user-comment graphs from Flickr[3] [32]. We present here results obtained with a file-provider dataset [2] which are representative of all obtained results.

[2] `http://www.delicious.com/`
[3] `http://www.flickr.com/`

	duration x of the reference period $[0, x[$				
	$x = 6$ hours	$x = 12$ hours	$x = 1$ day	$x = 3$ days	$x = 7$ days
number of $\perp$-nodes (peers)	82,372	122,817	160,159	356,197	705,634
number of $\top$-nodes (files)	1,474,048	2,060,530	2,456,205	3,938,639	5,703,258
number of links in E	2,764,424	4,259,764	5,634,865	11,851,292	22,334,912
number of links in $E_\perp$	11,792,614	27,519,054	54,182,976	273,674,542	1,045,199,202

Table 1: Number of $\perp$-nodes (peers), $\top$-nodes (files), and links in the bipartite graph G and in its $\perp$-projection $G_\perp$, for five different reference period durations x.

We first describe this real-world dataset. We show that its amount of internal links is high, which ensures the relevance of predicting internal links. We then discuss appropriate parameters for our experimentation for both our method and the collaborative filtering one. We present the results of our experimentations in Section 7.

6.1 File-provider dataset

We use for our experiments a measurement of a *peer-to-peer* file exchange network [2]. It consists of a recording of messages received and sent by a large *eDonkey* server during almost 10 weeks, with queries from users for files, and answers from the server indicating which users provide which files. We focus here on file-provider relations, leading to a set $D = \{(t_i, u_i, v_i)\}$ of triplets indicating that the server pointed peer u_i as a provider for file v_i at time t_i.

Using the formalism described in Section 4, each triplet corresponds to a link between a $\perp$-node (a peer) and a $\top$-node (a file) at time t_i. For two given timestamps x and y we consider:

- the reference period $[0, x[$ and the corresponding reference graph $G = (\perp, \top, E)$ induced by links observed from the beginning of the measurement (time 0) to time x, and
- the prediction period $[x, y[$ and the corresponding set of links $E' \subseteq (\perp \times \top) \setminus E$ added to G between x and y.

The $\perp$-projection $G_\perp$ of G is the graph in which two peers are linked if they provide one or more files in common.

Basic features of the reference graph G and its projection $G_\perp$ are presented in Table 1, for different reference period durations x. Notice that the number of $\top$-nodes (files) is much larger than the number of $\perp$-nodes (peers), which is mostly due to the fact that we consider only peers which provide at least one file (most only download files). Notice also that the number of links in the $\perp$-projections is huge. This is due to the fact that each $\top$-node v induces $\frac{|N(v)|\times(|N(v)|-1)}{2}$ links in the $\perp$-projection; see [25] for a discussion of this. To this regard, it is crucial that the methods considered in this paper need not store $G_\perp$, unlike most other approaches.

6.2 Amount of internal links

In order to gain more insight on the dynamics of the considered data, let us consider the number $|E'|$ of new links appearing during the prediction period $[1, y[$, for $y = 2, ..., 55$ days, for a reference graph G obtained with a reference period $[0, 1$ day$[$, presented in Figure 6 (left). The number of new links grows rapidly with the size of the prediction period, showing that many new links appear between nodes of the reference period, even after a long time. As one may expect, though, the number of new links grows faster during the first few days.

The fraction of internal links among these new links is presented in Figure 6 (right). It is very high, above 35%, for prediction periods of up to 10 days, with a maximal at almost 45%. The

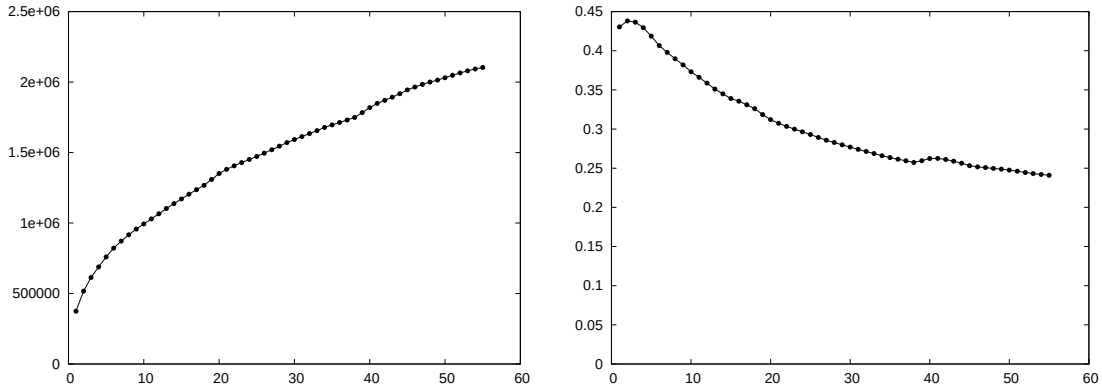


Figure 6: Number of new links (left) and fraction of internal links among them (right) as functions of the prediction period duration (horizontal axis, in days), with a reference period $[0, 1 \text{ day}[$ of one day.

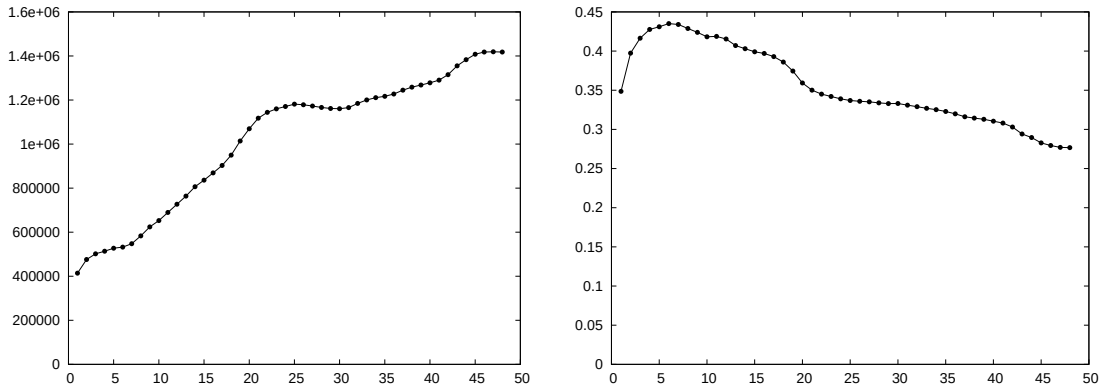


Figure 7: Number of new links (left) and fraction of internal links among them (right) as functions of the reference period duration (horizontal axis, in hours), with a prediction period $[x, x + 15 \text{ days}[$ of 15 days.

fraction of internal links decreases as the length of the prediction period grows, but it remains above 25% for prediction periods of up to 50 days.

Let us now observe how the number of new links $|E'|$ and the fraction of internal links among them evolves as the duration of the reference period grows. We consider reference periods $[0, x[$, for $x = 1, 2, ..., 48$ hours, and for each x we consider the 15 day prediction period $[x, x+15 \text{ days}[$; and present the number of new links in Figure 7 (left). We observe that it grows rapidly with the reference period duration x.

The fraction of internal links among these new links is presented in Figure 7 (right). It increases from 35% to 45% for reference periods from 1 to 6 hours. After this it decreases slowly but remains above 28% for reference periods of up to 48 hours.

These statistics show that the fraction of internal links is very high in the considered dataset, even for long reference and prediction periods, which is also true for other datasets we tested. This motivates our approach of focusing on this special class of links.

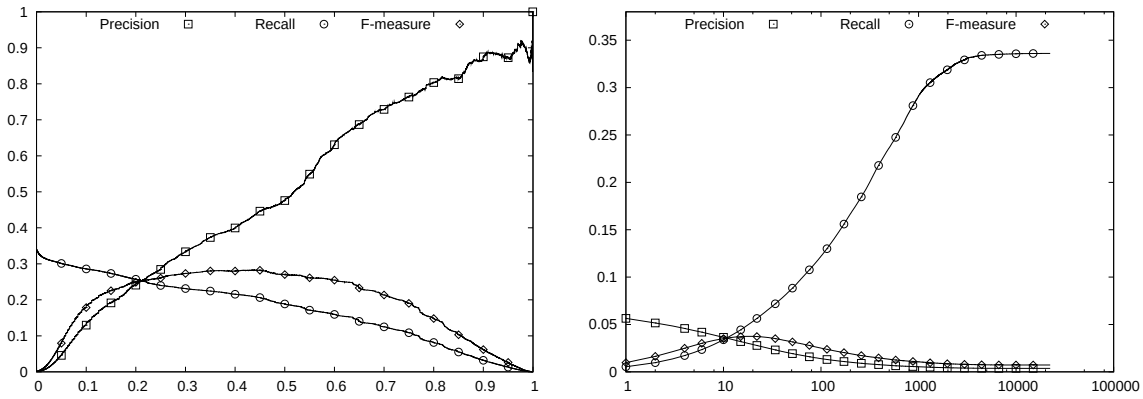


Figure 8: Figure 8: Performances of two prediction methods (left: *internal link prediction*; right: collaborative filtering with $k = 50$) for the reference period $[0, 1$ day$[$ of one day, the prediction period $[1, 16$ days$[$ of 15 days, and the *Jaccard* weight function. We plot the precision, recall, and F-measure (vertical axis), defined in Section 4, as functions of the threshold τ (left), and the number N of predicted links per node (right).

6.3 Parameters for prediction methods

The performances of link prediction methods depend on various parameters. We explore in depth in the next section (Section 7) the impact of the reference and prediction periods durations, as well as the impact of the weight functions. Even when these parameters are given, though, other parameters play a role: the weight threshold τ for *internal link prediction* and the values of N and k for collaborative filtering prediction. Exploring all possible values for all these parameters and their combinations is intractable in practice, and would actually have limited interest here as we are mostly concerned with *qualitative* results. We explain in this section how we choose values for these parameters for our experiments while avoiding extensive exploration of all possible values.

Internal link prediction

As illustrated in Figure 8 (left), the performances of *internal link prediction* for given reference and prediction periods and a given weight function depend on the weight threshold τ. If $\tau = 0$ then all possible internal links are predicted, which corresponds in this example to 33% of all appearing links. However, many of these links do not actually appear, and so the corresponding precision is almost zero. Instead, if a very high threshold is used then only few internal links are predicted, and so the obtained recall is almost zero. However, most of these few links do appear, which corresponds to a precision of almost 100%. More generally, the precision increases with the threshold value, and the recall decreases. The F-measure which captures a tradeoff between the two reaches its maximal value of 0.28 for $\tau = 0.4$ in this example.

To avoid taking into account the impact of the threshold τ on the *internal link prediction* method, we will select in the experiments of Section 7 the value of τ which maximizes the F-measure. For instance, when studying the impact of the prediction period duration x for a given reference period (Section 7.1) we will plot the maximal value of the F-measure as a function of x (Figure 9, left).

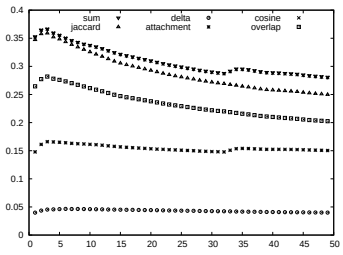
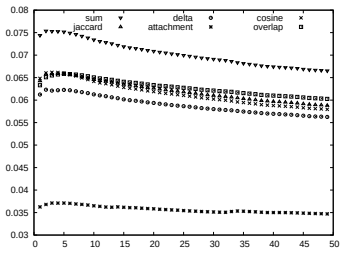
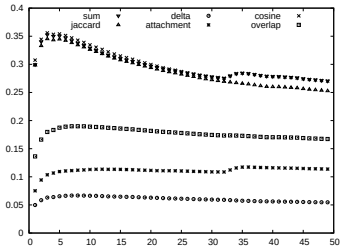
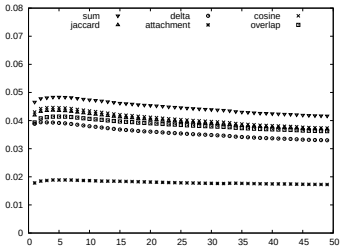
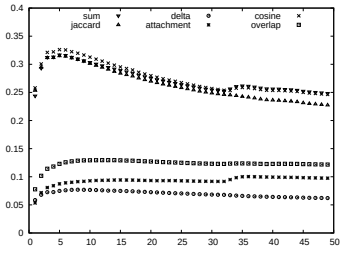
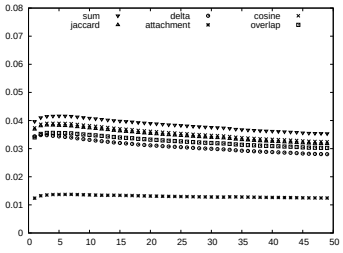
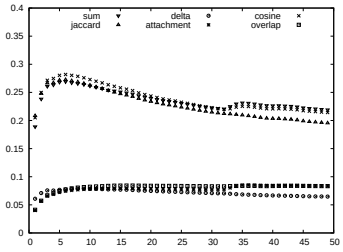
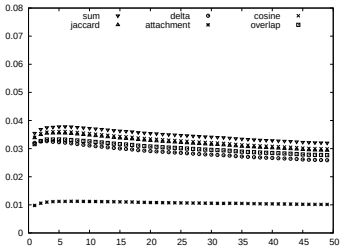
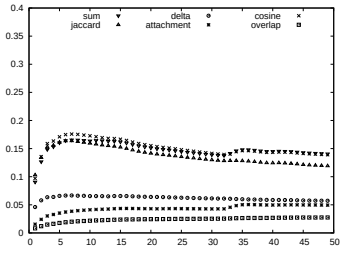
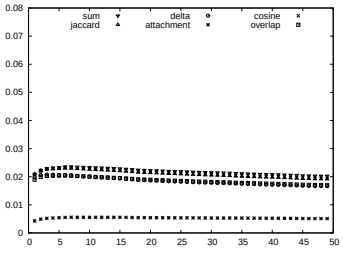
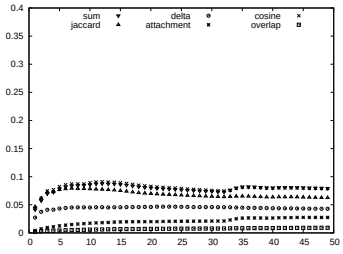
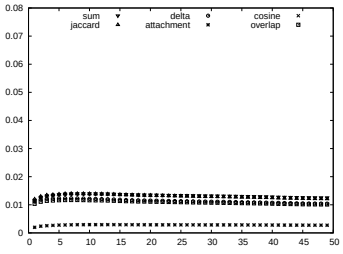
reference period	internal link prediction	collaborative filtering
1 hour		
6 hours		
12 hours		
1 day		
3 days		
7 days		

Figure 9: Maximal value of the F-measure (vertical axis) as a function of the prediction period duration (horizontal axis, in days) for both link prediction methods with different weight functions. In order to help comparison between different reference period durations, we used the same scale for the vertical axes. Notice however that the scales are not the same for the two methods because otherwise the collaborative filtering plots would hardly be readable.

Collaborative filtering

As explained in Section 5.2, the collaborative filtering method depends on a parameter k which is the number of similar neighbors considered for each node. We have experimented the performances of the collaborative filtering algorithm for three values of k: 10, 50 and 100. Results indicate that the precision is slightly better for small values of k, but recall is best for large values of k. The maximal F-measure was obtained for $k = 50$, therefore we will use this value in all our experiments in the following.

The other parameter, N, is the number of links predicted for each node. It also has a strong impact on the performance of the collaborative filtering method. Figure 8 (right) shows that the precision decreases and the recall increases when N increases.

Again, to avoid taking into account the impact of N on the performances of the collaborative filtering method, we will select in the following the value of N which maximizes the F-measure.

7 Experimental results

In this section, we study the performances of our approach for link prediction and compare it to the collaborative filtering approach in the experimental framework described above. We first explore the impact of the prediction period duration, which allows us to choose a relevant value for this parameter for the rest of our comparisons. We then study the impact of the reference period duration, and again select a relevant value for this parameter. Finally, we compare the performances of the different weight functions for these parameters.

7.1 Impact of the prediction period duration

In order to study the impact of the prediction period duration we consider several reference periods $[0, x[$ (from $x = 1$ hour to $x = 7$ days), and several prediction periods $[x, y[$ ($y = x + 1$ day, $y = x + 2$ days, ..., $y = x + 49$ days). We then compute, for each considered reference period duration, the maximal value of the F-measure observed over all values of the threshold τ (for *internal link prediction*) and all values of N (for collaborative filtering), and plot it as a function of the prediction period duration, as explained in Section 6.3.

Results are presented in Figure 9. The following key facts appear clearly:

- all plots have the same global shape (a fast increase followed by a slow decrease or steady regime), although their amplitude decreases when the reference period duration increases (we will deepen this in the next section);
- different weight functions give different results, which we deepen in Section 7.3;
- in most cases, and for all weight functions which perform well, *internal link prediction* surpasses significantly collaborative filtering (notice the different scales for the vertical axes for the two methods).

Finally, a prediction period of 15 days gives good results, and is representative of a wide range of prediction period durations for all reference period durations and weight functions. We will therefore use this prediction period duration in all the following.

7.2 Impact of the reference period duration

In order to investigate the impact of the reference period duration $[0, x[$, we vary its duration x for $x = 1, 2, ..., 48$ hours, and we use the prediction period $[0, x + 15$ days$[$ of 15 days, as

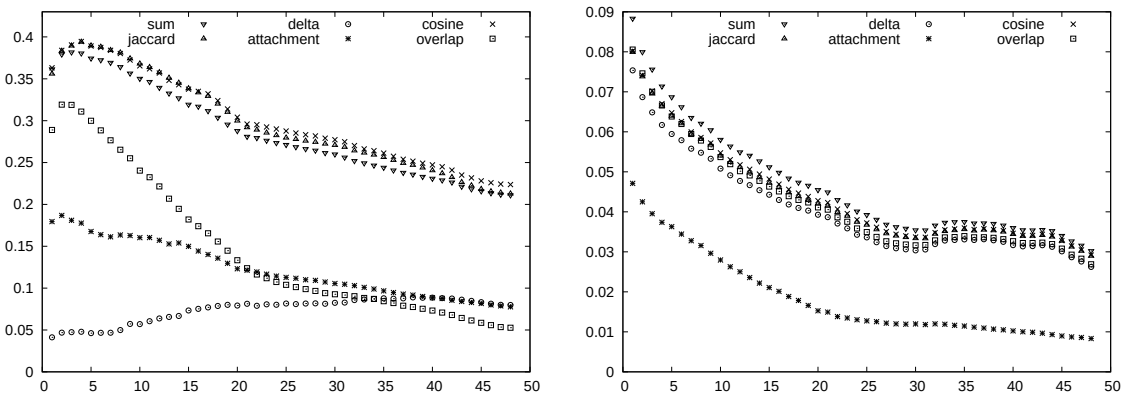


Figure 10: Evolution of the maximal F-measure (vertical axis) as a function the reference period duration (horizontal axis, in hours) for the different weight functions. Left: *Internal link prediction*. Right: collaborative filtering. The prediction period duration is 15 days in all cases.

explained in the previous section. We do not consider reference periods longer than 48 hours days, because, as we can see in Figure 9, longer reference periods lead to poorer performances. We compute for all cases the maximal value of the F-measure observed over all values of the threshold τ (for *internal link prediction*) and all values of N (for collaborative filtering) and plot it as a function of the reference period duration, as explained in Section 6.3.

Results are presented in Figure 10. The following key facts appear clearly:

- overall, the maximal F-measure decreases with the size of the reference period (except for very short reference periods with *internal link prediction*);
- different weight functions give different results, which we deepen in next section;
- in most cases, and for all weight functions which perform well, *internal link prediction* surpasses significantly collaborative filtering.

Finally, a reference period of 1 day gives good results, and is representative of a wide range of reference period durations. We will therefore use this reference period duration in all the following.

7.3 Impact of the weight function

In this section, we observe the impact of the weight function on both considered prediction methods. As explained in previous sections, we use reference period $[0, 1$ day$[$ and prediction period $[1, 16$ days$[$, which are representative of wide ranges of values for these parameters. We then compute the precision and recall for all possible values of the threshold τ for *internal link prediction* and all possible values of N for collaborative filtering; we plot the obtained precision as a function of the obtained recall in Figure 11.

A first important observation is that the weight functions considered clearly split into two classes regarding the performances of *internal link prediction* (Figure 11, left): sum, Jaccard and cosine reach very high values of precision, and are also able to reach very good compromises between precision and recall (like a precision of 50% and a recall of 20%); instead, delta, overlap and attachment lead to poor performances of *internal link prediction*. No such behavior is observable for collaborative filtering (Figure 11, right): all weight functions lead to very similar results except attachment which performs worse than the others.

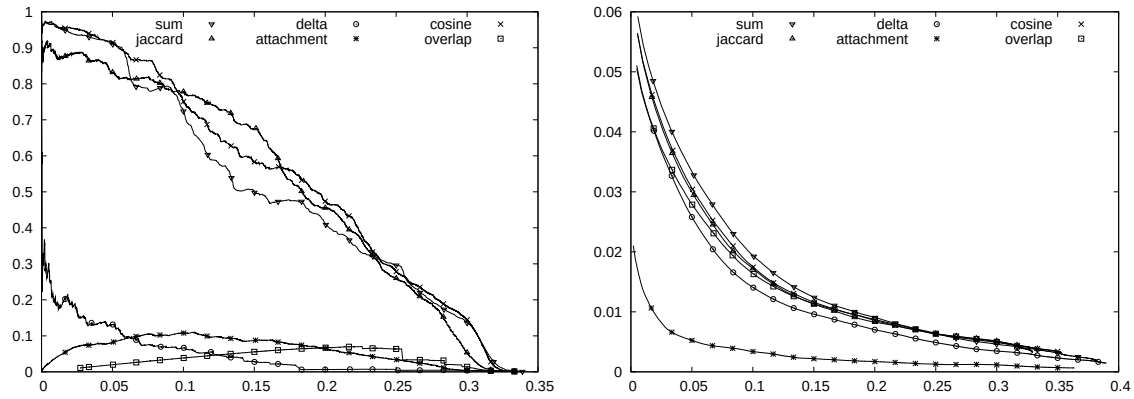


Figure 11: Precision (vertical axis) as a function of recall (horizontal axis), for a 1 day reference period $[0, 1[$ and a 15 days prediction period $[1, 16$ days$[$, for all weight functions. Left: *internal link prediction*; right: collaborative filtering. Each point corresponds to the precision and recall obtained for a given value of τ or N.

8 Conclusion

In this paper, we introduce a new class of links in bipartite graphs, which we call *internal links*, and propose a method which uses them for solving the link prediction problem. We evaluate the relevance of this method by comparing it to a classical collaborative filtering approach and perform experiments on various datasets, which show that our method performs very well. We present in details the results obtained for a file-provider graph in order to illustrate how results depend on various parameters, including which weight function is used for projection.

Our link prediction method has the following advantages. First, it performs very well, much better than a collaborative filtering approach, where no other method was previously available. Moreover, our method is purely structural: it relies on the identification of a specific kind of links which will probably appear in the future; this gives much insight on the properties of the underlying dynamics. Finally, the use of weight functions allows to tune the method in order to reach target tradeoffs in the quality of the prediction: one may use small thresholds to have excellent precision at the cost of a poorer recall, and conversely.

Our work may be extended in several ways. In particular, other (maybe more specific) weight functions may be introduced and tested. One may also predict internal links that induce *only* links with weight above the threshold (inducing *one* such link is sufficient in our current algorithm), or use both $\top$- and $\bot$-projections (our current algorithm only uses the $\bot$- one). There is therefore room for improving the method and its results.

Likewise, it would be interesting to conduct more experimentations and compare results on different datasets. Comparing our method with others, in particular machine learning approaches like the one presented in [6] is also appealing. Last but not least, our work calls for the development of link prediction methods for *external* links (those links which are not internal), which may be done with such methods or by modifying ours.

Another interesting direction would be to modify our approach in order to perform recommendation. As already explained, link prediction and recommendation are quite different problems, but they are strongly related. Just like we adapted collaborative filtering for link prediction in bipartite graphs, one may adapt our method and evaluate its relevance for recommendation.

Finally, we think that the notion of *internal links* introduced in this paper is fundamental and may be used as a building block in a much wider scope, in particular analysis of bipartite graphs. Although different, it is close to the notion of *redundancy* proposed in [25], which is one of the main statistics currently used for comparing real-world bipartite graphs and random ones. The fraction of internal links in any bipartite graph and similar statistics based on internal links may be used for this same purpose, and have significant advantages over redundancy (in particular, it is not a local measure, and is related to the graph dynamics). We consider this as one of the main perspectives of our work.

Acknowledgements

We warmly thank Nasserine Benchettara, Cécile Bothorel, Ludovic Denoyer, and Rushed Kanawati for their comments on preliminary versions. This work is supported in part by the French National Research Agency under contracts MAPE ANR-07-TLCOM-24 and DynGraph ANR-10-JCJC-0202.

References

[1] Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 2003.

[2] Frédéric Aidouni, Matthieu Latapy, and Clémence Magnien. Ten weeks in the life of an eDonkey server. In *Proceedings of the Sixth International Workshop on Hot Topics in Peer-to-Peer Systems (Hot-P2P'09)*, 2009.

[3] Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 1997.

[4] A. L. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 2001.

[5] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press, 1998.

[6] Nesserine Benchettara, Rushed Kanawati, and Celine Rouveirol. Supervised machine learning applied to link prediction in bipartite social networks. *Social Network Analysis and Mining, International Conference on Advances in*, 2010.

[7] Nesserine Benchettara, Rushed Kanawati, and Céline Rouveirol. A supervised machine learning link prediction approach for academic collaboration recommendation. In *Proceedings of the fourth ACM conference on Recommender systems*, 2010.

[8] John Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

[9] John Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002.

[10] Prabhakar Raghavan Christopher D. Manning and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[11] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 2008.

[12] Mukund Deshpande and George Karypis. Item based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 2004.

[13] D. M. Dunlavy, T. G. Kolda, and E. Acar. Temporal Link Prediction using Matrix and Tensor Factorizations. *ACM Transactions on Knowledge Discovery from Data*, 5(2), 2011.

[14] Karypis George. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, 2001.

[15] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 2001.

[16] Olaf Görlitz, Sergej Sizov, and Steffen Staab. Pints: Peer-to-peer infrastructure for tagging systems. In *Proceedings of the Seventh International Workshop on Peer-to-Peer Systems, IPTPS*, 2008.

[17] Jean-Loup Guillaume, Matthieu Latapy, and Stevens Le-Blond. Statistical analysis of a P2P query graph based on degrees and their time-evolution. In *Proceedings of the 6-th International Workshop on Distributed Computing (IWDC'04)*, 2004.

[18] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *Proceedings of SDM 06 workshop on Link Analysis, Counterterrorism and Security*, 2006.

[19] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, 1999.

[20] Zan Huang. Link Prediction Based on Graph Topology: The Predictive Value of Generalized Clustering Coefficient. *Workshop on Link Analysis Dynamics and Static of Large Networks (LinkKDD'06)*, 2006.

[21] Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Proceedings of the Joint Conference on Digital Libraries (JCDL05). ACM*, 2005.

[22] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 1997.

[23] Jérôme Kunegis and Sahin De Luca, Ernesto W. Albayrak. The Link Prediction Problem in Bipartite Networks. In *Proc. Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems*, pages 380–389, 2010.

[24] Jérôme Kunegis, Damien Fay, and Christian Bauckhage. Network Growth and the Spectral Evolution Model. In *Proc. Int. Conf. on Information and Knowledge Management*, pages 739–748, 2010.

[25] Matthieu Latapy, Clémence Magnien, and Nathalie Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social Networks*, 2008.

[26] Stevens Le-Blond, Jean-Loup Guillaume, and Matthieu Latapy. Clustering in P2P exchanges and consequences on performances. In *Proceedings of the 4th International Workshop on Peer-To-Peer Systems (IPTPS'05)*, 2005.

[27] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the twelfth international conference on Information and knowledge management(CIKM '03)*, 2003.

[28] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 2003.

[29] M. E. J. Newman. Clustering and preferential attachment in growing networks. *Physical Review Letters E*, 2001.

[30] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. In *Proceedings of the National Academy of Sciences of the United States of America*, 2002.

[31] Joshua O'Madadhain, Jon Hutchins, and Padhraic Smyth. Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explorations Newsletter*, 2005.

[32] Christophe Prieur, Dominique Cardon, Jean-Samuel Beuscart, Nicolas Pissard, and Pascal Pons. The stength of weak cooperation: A case study on flickr. *Computing Research Repository, CoRR*, 2008.

[33] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Collaborative Work Conference*. Proceedings of the Association of Computing Machinery, 1994.

[34] Paul Resnick and Hal Varian. Recommender systems. *Communications of the ACM*, 1997.

[35] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 1975.

[36] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1986.

[37] Tomasz Tylenda, Ralitsa Angelova, and Srikanta Bedathur. Towards time-aware link prediction in evolving social networks. In *Proceedings of the 3rd Workshop on Social Network Mining and Analysis (SNA-KDD '09)*, 2009.

[38] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[39] Chao Wang, Venu Satuluri, and Srinivasan Parthasarathy. Local probabilistic models for link prediction. *Data Mining, IEEE International Conference on*, 2007.

[40] D. J. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 1998.

[41] Sholom M. Weiss, Sholom M. Weiss, Nitin Indurkhya, and Nitin Indurkhya. Lightweight collaborative filtering method for binary encoded data. In *Proceedings of the Fifth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2001.