A multi-class approach for ranking graph nodes: models and experiments with incomplete data

Gianna M. Del Corso[a,*], Francesco Romani[a]

[a] *Università di Pisa, Dipartimento di Informatica, Largo Pontecorvo, 3, 56127 Pisa, Italy*

Abstract

After the phenomenal success of the PageRank algorithm, many researchers have extended the PageRank approach to ranking graphs with richer structures in addition to the simple linkage structure. Indeed, in some scenarios we have to deal with networks modeling multi-parameters data where each node has additional features and there are important relationships between such features.

This paper addresses the need of a systematic approach to deal with multi-parameter data. We propose models and ranking algorithms that can be applied to a large variety of networks (bibliographic data, patent data, twitter and social data, healthcare data). We focus on several aspects not previously addressed in the literature: (1) we propose different models for ranking multi-parameters data and a class of numerical algorithms for efficiently computing the ranking score of such models, (2) we analyze stability and convergence of the proposed numerical schemes and we derive a fast and stable ranking algorithm, (3) we analyze the robustness of our models when data are incomplete. The comparison of the rank on the incomplete data with the rank on the full structure shows that our models compute consistent rankings whose correlation is up to 60% when just 10% of the links of the attributes are maintained.

Keywords: Link Analysis, Models for Ranking Graph Nodes, Missing Links, Web Matrix Reducibility and Permutation

1. Introduction

Ranking algorithms are essential tools for searching in large collections of data and they are becoming more and more important as the amount of available data gets bigger and richer. Following the introduction and the success of PageRank and other ranking algorithms [8, 17], researchers have extended similar techniques to a multitude of domains [4, 5, 10, 13, 18].

With the advent of the semantic web, data containing many types of features and relationships are becoming common. Algorithms taking advantage of such additional information are needed and sought. Many analytical techniques have been proposed to better understand these data and their properties. Particularly important are ranking algorithms that evaluate objects on the basis

*Corresponding author

Email addresses: gianna.delcorso@unipi.it (Gianna M. Del Corso), romani@di.unipi.it (Francesco Romani)

of ranking functions measuring some characteristics of the objects. With such functions any two objects of the same type can be sorted in a partial order and compared either qualitatively or quantitatively. Ranking algorithms are widely applied in different network settings to get an overall view of the data.

In this paper we consider the setting in which the data consist of a collection of linked items, where each item has a set of additional attributes (features). In this setting it is natural to assume that the ranking of items with common attributes are mutually influenced. Many important problems are instances of this general framework.

In *bibliographic ranking* items are scholarly papers with the linkage structure provided by their citations. To each paper it is natural to associate features such that: authors, publication venue, subject classification and so on. The ranking of scientific papers on the basis of the received citations and publication venue has become an increasingly popular topic since the late 80's due to its importance for recruiting, promotions, and funding.

In *patent data analysis* items are patents linked by the citations to older patents. To each patent we can associate inventors, firm, examiner, technologies, *etc.*. Studies in marketing science utilize patents to examine different aspects of innovation: to understand knowledge flow within and across firms, to describe how knowledge flow influences the success of innovation, and to identify antecedents and outcomes of product innovation. An example of this line of research is [30] where the author shows that the number of patents owned by a firm (its patent count) correlates with R&D expenditure and represent a specific type of resource (intellectual property) the firm can use in various market processes.

Other examples of multi-parameter networks are: *social or twitter graphs*, where we have information about status, geographical location, education, *etc.* of users, and *healthcare data networks* where we have information on patients, doctors, treatments, diseases, *etc.*.

With a little abuse of notation in the following we use the term "multigraph" to denote this kind of relationships between items and features, while other authors identify this kind of graph with as heterogeneous information networks [28]. As shown by the above examples, the multigraph framework encompasses many different applications in which one has to compare different entities on the basis of their attributes and relationships. For this reason our results should be of interest for researchers in the information retrieval community as well as economists and people interested in the analysis of social networks.

In this paper we describe different models for representing the multigraph structure of a network. We analyze different techniques for assigning weights to features and to use these weights in the ranking process. These weights capture the importance that each link confers to the linked object. We then build a fast and stable numerical method for computing the ranking score according to our models. The proposed algorithm is obtained by combining two non-stationary methods (BCGStab [23] and TFQMR [23]) and a final phase of iterative refinement.

We perform many tests on two large datasets of patent data extracted from the US patent office: the first dataset consists of all the patents granted in the period 1976–1990 (roughly 2.5 Million patents), and the second of those issued between 1976 and 2012 (almost 8 Million patents). The experiments aim at understanding the differences between the various models and the role of the parameters involved in the algorithm. We also compare the results with those returned by PageRank and the ordering induced by the simple citation count.

We briefly investigate also the robustness of our models when data are incomplete and unrecoverable. In this setting our goal is to use all the information available without advantaging players (items or features) with more complete data respect to those where some information is missing. We treat unknown values as zeroes, since often we cannot distinguish between missing

(not available) or absent (not existent) features. This is the only viable choice when the missing data are unrecoverable and is the strategy implemented in patent repositories and in citation databases such as *Scopus, Mathscinet* where, for example, a citation is not attributed to anyone when the name of an author has been misspelled.

Following an established approach [36, 16], we evaluate the robustness of our ranking schemas on incomplete data by randomly removing features from items with an assigned probability. Our experiments show that, even removing up to half of the features, the ranks provided by our algorithm highly correlate to the ranks computed on the complete data. As expected, as more and more features are removed, the ranks converge to the rank obtained using only the linkage structure.

Finally, we tested the robustness of our models with respect to the granularity of the features. For example if we are dealing with bibliographic data we can group papers into subject classes where the granularity can be subject macro areas (Math, Computer Science, *etc.*) or finer classifications (Algebra, Number Theory, Calculus, Algorithms, Data Bases, *etc.*). In this context it is desirable that, when using a finer classification, the sum of the ranks of topic A subtopics is close to A's rank computed using the coarser classification. Experiments with the US patent dataset show that most of our models have such desirable feature.

The paper is organized as follows. In Section 1.1 we formally introduce the problem considered in the paper. In Section 1.2 we motivate our study and connect the techniques and the algorithm we propose with the existing literature. In Section 2 we briefly present some models discussing how extra information and features can be added to the citation structure to improve ranking and possible weighting criteria for such features. In our models the ranking is obtained approximating the Perron vector of a suitable stochastic matrix.

In Section 3 we discuss different ways for approximating the Perron vector showing that it can be obtained computing the solution of a linear system. In Section 4 we discuss different methods for the numerical solution of such linear system and we describe the databases used for the experiments. In Section 5 we report an extensive numerical testing to compare the different models in terms of convergence for missing data and consistence for class aggregation. Section 6 contains the conclusion and some discussions about possible improvements of the models.

1.1. Preliminaries and notations on multigraphs

In this paper we consider a multigraph as described by a directed graph $G = (V, E)$ and two mapping functions, one for the nodes $\tau : V \to \mathcal{A}$ and one for the edges $\phi : E \to \mathcal{R}$. Each node $v \in V$ belongs to a particular type $\tau(v) \in \mathcal{A}$ and each edge $e \in E$ belongs to a particular type of relation $\phi(e) \in \mathcal{R}$. Functions ϕ and τ are such that if e_1 and e_2 are two edges, $e_1 = (v_1, v_2)$ and $e_2 = (w_1, w_2)$, with $\phi(e_1) = \phi(e_2)$, then $\tau(v_1) = \tau(w_1)$ and $\tau(v_2) = \tau(w_2)$. When $|\mathcal{A}| > 1$ and $|\mathcal{R}| > 1$ we say that the graph is a multigraph.

An example of multigraph is a patent network, where each patent has associated different features in the set $\mathcal{A} = \{$Patent, Technology, Firm, Examiner, Inventor and Lawyer$\}$. The different relation types are the edges between patents and firms, patents and examiners, patents and the set of inventors, and patents and lawyers. In addition each patent has outcoming edges to the patents cited in its technical description. Each kind of edge has a different semantic meaning: for example the connections between inventors and patents express intellectual property over the patents while the edges between patents and examiners represent the fact that a patent was granted by a particular examiner. Figure 1 shows the relations between the different features and the different kinds of nodes.

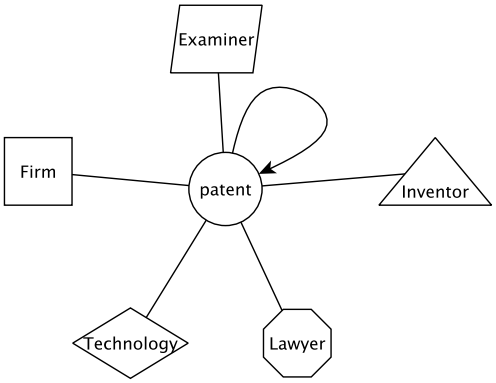

Figure 1: Schema of a patent graph. Each patent has a relation with other types of nodes.

To better understand multigraphs and the information contained and expressed by the different types of relations (edges) between nodes, we associate to the graph a model describing the interaction between items, features and the possible interactions between features. In this paper we define several models and compare them on the basis of a ranking function inspired by the PageRank algorithm. In particular, the original network schema is enriched by including in the model other information that can be derived from the relations between nodes, such as the network of co-inventors, or all the combinations between any two couples of features, i.e. firm-technology or examiner-inventor, *etc*. These enriched models allow to define a ranking function mapping objects to a real non-negative score representing the importance of the object. The rank accounts for all the information available and not only for the citation network, and allows to rank all types of nodes on the basis of the linkage structure of the enriched graph.

1.2. Motivations and Related work

Ranking algorithms are essential when searching in large collections of data, being either web pages, bibliographic items or even healthcare data. Recently, many ranking algorithms have been developed [8, 17, 20, 27, 31] that take advantage of the specific structure of the underlined graph. Also in the area of economics it is common practice to use ranking metrics for evaluating the performance of markets and country economies. Recently, [26] has proposed a ranking algorithm based on PageRank for patent data. Despite the whole information about patents is available from the USPO (US Patent office) only the citation structure has been considered in [26] and the multigraph structure of the patent graph, including information on firms inventors and technologies, has not been fully exploited. Since patents are often used to measure innovation of entrepreneurial activities [3, 11] a ranking schema taking into account all the features of patents can be used not only for evaluating the innovation of the patented idea or product, but also to evaluate firms or for portfolio management. This is the primary reason we tested our ranking algorithm on patent data, even if the techniques we present can be applied to any multigraph structure.

Comparing different ranking algorithms is a very difficult task when, as for this problem, no golden truth is available. In some cases it is possible to take a panel of volunteers and let them manually evaluate the data, but in most cases, either for the size of the data or for the expertise

required, this is not possible. For example, manually ranking patents requires a remarkable knowledge of the field and such expertise is not easy to find. Another difficulty in comparing ranking algorithms is that we can use the same data to discover different properties: in this case a direct comparison is not possible. For instance, if we want to evaluate scholars on the basis of their ability to work in a team, we will design a ranking function highly valuing the scholars with many coauthors, while if we are interested in scientific personal strength it is natural to normalize each publication by the number of coauthors. The resulting rankings will likely be incommensurable.

In this paper we propose a tunable ranking algorithm where by changing parameters we can accomplish different goals. In particular, the same algorithm can be used on different kind of data and for different purposes. One of the parameters is the model itself and another one is the weighting strategy. This is the major difference with previous ranking algorithms that are designed for specific networks and appear to be less tunable [27, 20, 28, 29, 33]. Together with the models we propose and analyze some weighting strategies. To change the ranking function one can implement other weighting strategies and incorporate them into the algorithm.

In the following we review other approaches for ranking multigraphs and compare them with our strategy. The problem of ranking "multigraphs", as informally defined above, has been recently considered in some specific domains. In [27] the multigraph is transformed into a layered graph with a layer for each feature. The ranks of each layer are computed independently and the final ranks are obtained with a linear combination of the layer ranks'. We believe the independent computation for each layer does not fully take advantage of the structure of the problem because the mutual interactions between the features are not taken into account. For the specific domain of bibliographic ranking, the PopRank algorithm powering Microsoft Academic Search introduced by Nie et al. [20] is a two phase extension of PageRank applied to typed multigraphs with different weights on the links. In particular the formula for the PopRank score combines with weight ε the so called "web popularity" that is a measure similar to the PageRank and with weight $1 - \varepsilon$ the popularity propagation factor of ongoing links. This factor is based on the importance of links pointing to an object and is computed with a learning based technique that automatically learn the popularity propagation factor for different types of links using the partial ranking of the objects given by *domain experts*. This ranking schema is very different from ours since PopRank uses an external human contribution and is therefore problem dependent and impossible to replicate on a different dataset.

A different approach for ranking multigraphs is the one that makes use of multilinear algebra and tensors for representing graphs with multiple linkages [18, 14, 1]. The tensor however does not contain the same information we use in this paper. For example, if we are dealing with bibliographic data, our models use the full author list for each paper, while the tensor only records the number of common authors between each pair of papers. Hence it does not allow to obtain a score for all the features such as authors or journals, and hence it is not possible to compare its results with those provided by our algorithm.

Sun and al. [28, 29] in the context of a bi-typed network (for example a bipartite bibliographic graph with only authors and conference venues) or star-typed networks (for example a bibliographic graph where we have papers and all the other features such as authors, conference venues, terms, are linked via papers) propose a ranking schema combined with clustering, where the clustering algorithm improves the ranking and vice-versa. One of the ranking function proposed is similar to ours but applies only to the simpler graphs described above with only two types of nodes. In [34] the authors, still in the context of bibliographic data, proposed a model similar to one of our models, namely the `Simple Heap` model (5). It mainly differs from ours

for the weighting strategy and the use of a non-static model. However, we consider an enriched structure with a complete set of relations between features. For example in the contest of bibliographic data we enrich the graph adding weighed links between authors, journals, and subject classification.

In previous papers from the same authors [5, 13, 6] a model is introduced in the context of bibliographic data that is similar to one of the models (the one we called `Stiff model`) of this paper. In particular in [5] an integrated model for ranking scientific publications together with authors and journals was presented. In that context, particular weighting strategies were implemented [6] and an exponential decay factor was introduced [13] to take into account aging of citations, i.e. the fact that if an old paper is not cited recently its importance should fade over the time. In this paper we further generalize the original ideas introducing several models and other classes making the model suitable also for ranking other multi parameters data (patents, healthcare, social data *etc.*). The new models are more adequate for example to handle updating of the datasets that can be done at a lower cost than in the `Stiff` model. In addition, in this paper the weighting strategies are problem independent, while in the previous papers they were designed ad hoc for dealing with bibliographic items.

Another contribution of this paper is the investigation of adequate numerical techniques to compute the ranking score. In particular, in Section 3 we show how the computation of the ranks relies upon the solution of a structured linear system and in Section 4 we discuss and compare the different algorithms that can be used to solve that system. Dealing with big data requires indeed particular care in the choice of the numerical methods used in the algorithms that should be stable and fast. The final algorithm (Procedure `SystemSolver` in Section 4) has been chosen on the basis of several tests aiming to validate its properties of convergence and stability. A similar analysis, for the ranking problem, has not been done in the literature, and often even methods requiring matrix manipulations [29] or spectral algorithms [34] miss to analyze this important aspect.

Another contribution of the paper is a preliminary analysis of the robustness of the algorithms in the presence of missing data. The problem of dealing with missing data is by no means unique to ranking algorithms. For example, in the field of computer vision, some data my be missing due to the presence of shadows or occlusions in the image. In [21, 9] this problem is addressed by approximating a matrix with unknown entries as the product of two low rank matrices.

Many techniques have been developed to deal with incomplete data and to make it possible to use corrupted or incomplete dataset. A common practice– and the easiest to apply– is to use only the items with complete information discarding those with incomplete data [22]. This is a rather drastic approach especially when a large portion of data is incomplete. As an alternative, researchers have proposed to fill in a plausible value for the missing observations. Among statisticians distributional models for the data, such as maximum likelihood [19, 24] and single or multiple imputation [24, 25], have been developed to replace non ignorable missing data. The goal of this paper is not however to study the preprocessing of data for recovery missing features. This topic would require adequate models and techniques [19, 22] to recover data and fill in the missing entries. In this paper we are only interested in quantifying how the ranking score is affected when some of the data are missing (completely) at random[1]. To this end we assume that a missing entry corresponds to a zero value in the linkage structure, such as is done in large

[1]The data are missing completely at random (MCAR) when the probability that a data is missing cannot depend on any other data in the model [2]. Alternative assumptions have been studied in the literature [19, 24, 2] such as the Missing at Random (MAR) or the Not Missing at Random (NMAR) cases.

bibliographic databases such as *Scopus*, *dblp*, or even the web when to a broken link we do not associate any link. We are aware that replacing a missing value with a zero is not a good choice when the data do not have homogeneous attributes [37], but in the case of bibliographic data, patent data or other networks fitting into the model of Figure 1, the set of the features is homogeneous. For instance, any paper has at least an author, a publication venue, *etc.* Adding and removing links at random is a common practice when evaluating performance of ranking algorithms on large social networks to measure the tolerance of ranking against spurious and missing links [16, 35, 36]. In Section 5.2 we show that also for our algorithm the ranking obtained with incomplete data highly correlates with the ranking obtained with the full dataset. Of course, our analysis does not rule out that in certain contexts an appropriate preprocessing for recovering missing data can improve the ranking provided by our algorithm.

2. Models

In Section 2.1 we present a link-based ranking for a simple citation graph. In Section 2.2 we enrich the graph with additional information (features) on the nodes.

2.1. The One-class model

In this model we have a citation matrix C, where $c_{ij} = 1$ if node i links to node j. There are many example of such matrices for example the web graph or the graph representing citations between scholar papers.

Following an idea similar to Google's PageRank [8], we assume that the importance p_j of node j is given by the importance of the nodes i citing j, scaled by d_i, the outdegree of i. The importance given by i is thus uniformly distributed among all the cited nodes, and the principle that the importance of a subject is neither destroyed nor created is respected.

Here and below, we denote by $\boldsymbol{e}$ the vector of appropriate length with all components equal to one. We denote by $\boldsymbol{e}_k$ the k-th column of the identity matrix of appropriate size. The size of vectors and matrices, if not specified, is deduced by the context. Given a vector $\boldsymbol{v} = (v_i)$ of n components, with the expression $\mathrm{diag}(\boldsymbol{v})$ we denote the $n \times n$ diagonal matrix having diagonal entries v_i, $i = 1, \ldots, n$.

Since nodes may have an empty set of links, the matrix C can have some null rows and in that case the corresponding outdegrees d_i are zero. To avoid divisions by zero we introduce a *dummy node*, numbered $n + 1$, that cites and is cited by all the existing nodes except itself. The new adjacency matrix of size $n + 1$, denoted by $\hat{C}$, has no null rows and is irreducible. The dummy node collects the importance of all the nodes and redistributes them uniformly to all its neighbors.

The outdegrees $d_i = \sum_j \hat{c}_{i,j}$ define the vector $\boldsymbol{d} = (d_i)$, that satisfies the equation $\boldsymbol{d} = \hat{C}\boldsymbol{e}$. Moreover, since $d_i \neq 0$ for all i, the matrix

$$P = (p_{i,j}) = \mathrm{diag}(\boldsymbol{d})^{-1}\hat{\mathrm{C}}$$

is row-stochastic, that is, $0 \leq p_{ij} \leq 1$, $\sum_j p_{i,j} = 1$.

A similar approach is used in the PageRank model where C is first normalized by row, and then a random jump probability α is introduced to make the matrix irreducible. In our model the probability to reach the dummy node is not the same for all nodes, but varies accordingly with the outdegree of each node.

The ranking or "importance" of each node is computed solving the following equation

$$\boldsymbol{x}^T = \boldsymbol{x}^T P, \qquad P = \mathrm{diag}(\hat{\mathrm{C}}\boldsymbol{e})^{-1}\hat{\mathrm{C}}. \tag{1}$$

Since the matrix $\mathrm{diag}(\mathrm{P}\boldsymbol{e})^{-1}\hat{\mathrm{C}}$ is nonnegative and irreducible, from the Perron-Frobenius theorem [32] there exists a unique vector $\boldsymbol{x} = (x_i)$ such that $x_i > 0$, $\sum_i x_i = 1$, that solves (1). We call $\boldsymbol{x}$ the *Perron vector* of P.

This model, that we call `One-class` has been introduced in [5]. It has been used to rank scientific papers [5] and patents [26]. In [7], assuming the citation matrix triangular, this model and the PageRank model are viewed as special cases of a family of Markov chain-based models.

2.2. Multi-class models

Often, beside the linkage structure we have additional information that can be profitably used in the ranking process. For example, to evaluate a paper we can use, besides the received citations, other information available such as the authors or the journal where the paper has been published. We now show that the mixing of all these ingredients (in this example authors, citations, journals) makes it possible to compute a better ranking for papers and, at the same time, a ranking score also for journals and authors.

The idea is to compute a ranking value for authors based on the quality of their papers and of the journals where the papers appeared. Journals can be evaluated as well using the information about the importance of the authors writing for that journal and of the papers published therein. This approach was first proposed in [5] and further extended in [13, 6]. We start with the original citation matrix C, then we add the information on the features of each item storing them in rectangular matrices. Examples of features are authors and journals if the items are scholar papers; or firms, inventors, technologies and lawyers if the items are patents. In general, we have f, $f = |\mathcal{A}| - 1$, rectangular binary feature matrices $F_1, \ldots, F_f$ (one for each feature) where entry (i, j) in F_k is different from zero iff item i has attribute j for feature k. For patent items, for example, we have the "inventorship feature matrix" storing information about the inventors of a patent, that is, entry (i, j) is nonzero if j is an inventor of patent i.

Given the $n_C \times n_C$ citation matrix C, the feature matrices F_k, for $k = 1, 2, \ldots, f$ where each F_k has size $n_C \times n_k$, and some weights α_{ij}, we can construct a block matrix A of size $N = n_C + \sum_{k=1}^{f} n_k$ in different ways leading to different models. Note that the size of A is equal to the number of items plus the number of attributes for each feature.

Once we have the block matrix A, we proceed as in the PageRank algorithm and we obtain ranks for both *items* and *attributes*. To compute the ranking score as in (1) we first force irreducibility in the underlying Markov chain and then normalize the resulting matrix to get a stochastic matrix P.

We now show that by varying the structure of the blocks combining the features and the strategy for forcing irreducibility we get four different base models. Combining these base models with different weighting strategies we obtain a total of 15 models summarized in Table 1.

`Stiff model` Each matrix F_k as well as the matrix C is embedded in a matrix with an additional row and column as follows

$$\hat{F}_k = \left[\begin{array}{c|c} F_k & \boldsymbol{e} \\ \hline \boldsymbol{e}^T & 0 \end{array}\right], \quad \hat{C} = \left[\begin{array}{c|c} C & \boldsymbol{e} \\ \hline \boldsymbol{e}^T & 0 \end{array}\right].$$

The matrix $\hat{A}$ is

$$\hat{A} = \begin{bmatrix} \hat{F}_1^T \hat{C} \hat{F}_1 & \hat{F}_1^T \hat{F}_2 & \cdots & \hat{F}_1^T \hat{F}_f & \hat{F}_1^T \\ \hat{F}_2^T \hat{F}_1 & \hat{F}_2^T \hat{C} \hat{F}_2 & \cdots & \hat{F}_2^T \hat{F}_f & \hat{F}_2^T \\ \vdots & \ddots & \ddots & \cdots & \vdots \\ \hat{F}_f^T \hat{F}_1 & \cdots & \ddots & \hat{F}_f^T \hat{C} \hat{F}_f & \hat{F}_f^T \\ \hat{F}_1 & \hat{F}_2 & \cdots & \hat{F}_f & \hat{C} \end{bmatrix}. \tag{2}$$

The matrix $\hat{A}$ is the adjacency matrix of a more complex multigraph respect to the one described by the schema in Figure 1. In fact all the possible relations between any pair of features is accounted for, meaning that the graph is complete and we have $(f+1)^2$ types of edges. The diagonal blocks are of the form $\hat{F}_k^T \hat{C} \hat{F}_k$ and contain the co-citations between features. For example, if F_k is the authorship matrix each entry of $\hat{F}_k^T \hat{C} \hat{F}_k$ accounts for the number of citations between any two authors. For off-diagonal blocks of type $\hat{F}_k^T \hat{F}_h$, for example when F_h is the paper-journal matrix[2], each entry accounts for how many papers an author has published on a given journal.

For the construction of the stochastic and irreducible matrix P we proceed as follows. We normalize by row each block of matrix $\hat{A}$, obtaining the stochastic and irreducible matrices $P_{i,j}$, for $i, j = 1, 2, \ldots, f, f+1$, where $P_{f+1,f+1}$ corresponds to the row normalization of $\hat{C}$. Then, given a row stochastic matrix of weights $\Gamma = (\gamma_{ij})$ with $i, j = 1, 2, \ldots, f, f+1$, we build matrix P as follows

$$P = \begin{bmatrix} \gamma_{1,1} P_{1,1} & \gamma_{1,2} P_{1,2} & \cdots & \gamma_{1,f+1} P_{1,f+1} \\ \gamma_{2,1} P_{2,1} & \gamma_{2,2} P_{2,2} & \cdots & \gamma_{2,f+1} P_{2,f+1} \\ \vdots & \vdots & \vdots & \\ \gamma_{f,1} P_{f,1} & \cdots & \gamma_{f,f} P_{f,f} & \gamma_{f,f+1} P_{f,f+1} \\ \gamma_{f+1,1} P_{f+1,1} & \cdots & \gamma_{f+1,f} P_{f+1,f} & \gamma_{f+1,f+1} P_{f+1,f+1} \end{bmatrix}. \tag{3}$$

We called this model `Stiff` because it lacks flexibility. In fact, if we add an attribute to a feature, we need to recompute not only the corresponding F_k and the matrices involving F_k in (2), but also renormalize each of the changed blocks. This approach was followed in [13, 6] for ranking papers, authors and journals[3]. In [13] some discussion about possible choices of the weights γ_{ij} are reported. Note that since the matrix of the weights Γ is stochastic and also the blocks P_{ij} are stochastic, matrix P describes a coupled Markov chain.

`Static model` This model differs from the previous because instead of adding a row and a column to each of the feature matrices, we add a dummy item to the whole matrix, and

[2] In the paper-journal matrix an entry (i, j) is nonzero if the paper i was published on journal j.

[3] In [13, 6] each block was normalized in a particular way because row normalization was not always well suited for that particular problem.

then weight each block with suitable parameters α_{ij}. We obtain the matrix

$$\hat{A} = \left[\begin{array}{ccccc|c} \alpha_{1,1} F_1^T C F_1 & \alpha_{1,2} F_1^T F_2 & \cdots & \alpha_{1,f} F_1^T F_f & \alpha_{1,f+1} F_1^T & \\ \alpha_{2,1} F_2^T F_1 & \alpha_{2,2} F_1^T C F_1 & \cdots & \alpha_{2,f} F_2^T F_f & \alpha_{2,f+11} F_2^T & \\ \vdots & \ddots & \ddots & \cdots & \vdots & \boldsymbol{e} \\ \alpha_{f,1} F_f^T F_1 & \cdots & \ddots & \alpha_{f,f} F_f^T C F_f & \alpha_{f,f+1} F_f^T & \\ \alpha_{f+1,1} F_1 & \alpha_{f+1,2} F_2 & \cdots & \alpha_{f+1,f} F_f & \alpha_{f+1,f+1} \tilde{C} & \\ \hline \multicolumn{5}{c|}{\boldsymbol{e}^T} & 0 \end{array}\right]. \quad (4)$$

We then normalize by row to get the stochastic irreducible matrix $P = \mathrm{diag}(\hat{\mathrm{A}}\boldsymbol{e})^{-1}\,\hat{\mathrm{A}}$. For this and the remaining models proposed in this section, whenever we add a new attribute to an existing feature we have to change only the matrix of the feature involved. Indeed, we do not need to build the matrix $\hat{A}$ explicitly but all the computation can be done using only the matrices F_k and C.

The next two models are designed for dealing with problems where the feature data is incomplete. For example in a bibliographic database where we only know the first author of each paper. In this case, we cannot expect to compute an accurate rank for authors, but still we would like to use the available author information to better rank papers. The structure of the blocks is now homogeneous among off-diagonal and diagonal blocks so that we can ideally consider all the features heaped in just a matrix F containing all the information on the different attributes. Matrix F has size $n_C \times n_t$, n_t being the total number of attributes, for example the sum of distinct authors, journals, *etc.* available. Since the Heap model can be used also with complete data we describe the model keeping the features distinct, knowing that the features can be squeezed in a unique matrix when the features classes are scarcely populated.

`Heap model` The `Heap` model differs from the `Static` model in the off-diagonal blocks. Blocks $F_k^T F_h$ are replaced by $F_k^T C F_h$. In the previous example where F_k was the paper-journal matrix and F_h is the paper-author matrix, the entry (i, j) of $F_k^T F_h$ is the number of papers author j has published on journal i, while the (i, j) entry of $F_k^T C F_h$ is the number of citations from papers written by author j to all papers published in journal i.

Assigning to each block a weight $\alpha_{i,j}$, we get the matrix $\hat{A}$

$$\hat{A} = \left[\begin{array}{ccccc|c} \alpha_{1,1} F_1^T C F_1 & \alpha_{1,2} F_1^T C F_2 & \cdots & \alpha_{1,f} F_1^T C F_f & \alpha_{1,f+1} F_1^T & \\ \alpha_{2,1} F_2^T C F_1 & \alpha_{2,2} F_1^T C F_1 & \cdots & \alpha_{2,f} F_2^T C F_f & \alpha_{2,f+11} F_2^T & \\ \vdots & \ddots & \ddots & \cdots & \vdots & \boldsymbol{e} \\ \alpha_{f,1} F_f^T C F_1 & \cdots & \ddots & \alpha_{f,f} F_f^T C F_f & \alpha_{f,f+1} F_f^T & \\ \alpha_{f+1,1} F_1 & \alpha_{f+1,2} F_2 & \cdots & \alpha_{f+1,f} F_f & \alpha_{f+1,f+1} \tilde{C} & \\ \hline \multicolumn{5}{c|}{\boldsymbol{e}^T} & 0 \end{array}\right].$$

To get the stochastic matrix P we just normalize $\hat{A}$ by row.

`Simple Heap model` In this model we assume that there is no interaction between features so

that cross-citations do not influence the rank.

$$\hat{A} = \left[\begin{array}{cc|c} O & \alpha_{1,2}\, F^T & \\ \alpha_{2,1}\, F & \alpha_{2,2}\tilde{C} & \boldsymbol{e} \\ \hline \multicolumn{2}{c|}{\boldsymbol{e}^T} & \end{array}\right], \tag{5}$$

where F is a matrix containing all the relations between items and attributes, *i.e.* $F = [F1, F2, \ldots, F_f]$. As already observed this model uses a simplified setting to deal with the case where we have incomplete data.

To check consistency of the proposed models, we have to prove that as $\alpha_{f+1,f+1} \to 1$ and $\alpha_{i,j} \to 0$, for all the other values of i, j, the rank obtained with these models converges to the rank obtained with the one-class model. This is however guaranteed because in all the models, for the limit value of $\alpha_{i,j}$, $\hat{A}$ collapses to a matrix of the form

$$\left[\begin{array}{cc|c} O & O & \\ O & \tilde{C} & \boldsymbol{e} \\ \hline \multicolumn{2}{c|}{\boldsymbol{e}^T} & \end{array}\right],$$

and the rank of the items is the same (up to a scaling factor) of the one obtained with the one-class model, while all the features will get an uniform score.

2.3. Weighting strategies

Weighting strategies play an important role in the tuning of the algorithm, since by varying them we can change the relative importance of features vs citations and consequently change the final ranking. We propose five different weighting strategies for our models, but not all strategies can be applied to each model, and for different models two weighting schemes may coincide after normalization of the matrix $\hat{A}$.

The simplest strategy is the `Uniform` (`U`) one, that corresponds to choosing $\alpha_{i,j} = 1$ for each $i, j = 1, \ldots, f + 1$. By adopting this weighting schema the contribution of each class (feature or citation) is valued in the same way, independently of its size. This approach appears adequate only when the sizes of each class are of the same order of magnitude, otherwise we are giving a bigger role in the determination of the ranking to scarcely populated classes.

For this reason, we also consider schemes that keep track of the size of each class. We have different choices.

`Dimension-based (D)` We set $\alpha_{i,j} = n_j/n_C$, and $\alpha_{i,f+1} = 1$. In this way we guarantee that the average value of the features are the same [13], and we do not advantage more populated classes respect to those less populated. The weights are the same for each block of columns.

`Double-Dimension-based (DD)` We have a symmetric weight matrix, setting $\alpha_{i,j} = \alpha_i \alpha_j$, where $\alpha_i = n_i/n_C$ is the normalized size of the i-th feature. In the case the citation matrix is much larger respect to the size of F_i, this scheme gives more importance to citations than to features.

models\weights	U	D	DD	H	HH
Stiff	Stiff-U	Stiff-D	-	-	-
Static	StaticU	Static-D	Static-DD	-	-
Heap	Heap-U	Heap-D	Heap-DD	Heap-H	Heap-HH
Simple-Heap	SHeap-U	SHeap-D	SHeap-DD	SHeap-H	SHeap-HH

Table 1: The 15 models obtained combining the basic models with the different weighting strategies.

Heap (H) We set $\alpha = (\sum_{k=1}^{f} n_k)/n_C$ for the first f blocks of columns, that is $\alpha_{i,j} = \alpha$ for $i = 1, \ldots, f+1$ and $j = 1, \ldots, f$ for the blocks in the last column we get $\alpha_{j,f+1} = 1$, for $j = 1, \ldots, f+1$. This weighting strategy is particularly suited for the Heap or Simple Heap model.

Double-Heap (HH) In this case the weights are not the same along the blocks of columns but defining $\alpha = (\sum_{k=1}^{f} n_k)/n_C$, we have $\alpha_{i,j} = \alpha^2$ for $i, j = 1, \ldots, f$, and the weights of blocks in the last column are $\alpha_{j,f+1} = \alpha$, and in the last row $\alpha_{f+1,j} = \alpha$. Moreover $\alpha_{f+1,f+1} = 1$. Also this scheme is particularly suited for the Heap or Simple Heap model since they have the same value in the upper left blocks. Assuming $\alpha < 1$ we are giving again more importance to citations when determining the ranking scores of the other nodes.

While it is always possible to apply an Uniform weight to each base model, it doesn't make sense to apply some of the weighting strategies to the Stiff or Static model. In fact the H or the HH weighting techniques make sense only when the structure of diagonal and off diagonal blocks is the same as in the case of the Heap or Simple-Heap model. Using the H or the HH weighting techniques in combination with the Heap model we can rewrite the matrix $\hat{A}$ in a more compact form collecting all the features in a unique matrix F. We get

$$\hat{A} = \left[\begin{array}{cc|c} \alpha_{1,1}\, F^T C\, T & \alpha_{1,2}\, F^T & \boldsymbol{e} \\ \alpha_{2,1}\, F & \alpha_{2,2}\tilde{C} & \\ \hline \multicolumn{2}{c|}{\boldsymbol{e}^T} & 0 \end{array}\right].$$

In Table 1 we summarize the fifteen full models obtained combining the four basic models, with the five weighting schemas.

3. Computation of the Perron vector

In all our models to compute the rank we have to solve an eigenvector problem involving a stochastic irreducible matrix. More precisely, we have to find the left Perron vector $\boldsymbol{x}$ such that $\boldsymbol{x}^T = \boldsymbol{x}^T P$, with P stochastic. We now show that the Perron vector can be computed as the solution of a linear system involving a matrix $\widehat{M}$, where

$$\widehat{M} = \begin{cases} \hat{C} & \text{if } f = 1 \\ \hat{A} & \text{if } f \geq 2. \end{cases}$$

separating the last row and column of $\widehat{M}$ we have

$$\widehat{M} = \left[\begin{array}{cc} M & \boldsymbol{u} \\ \boldsymbol{v}^T & 0 \end{array}\right],$$

where M has size $N \times N$ and $\boldsymbol{u}, \boldsymbol{v}$ are suitable N-vectors (for the Stiff models $\boldsymbol{u}, \boldsymbol{v}$ are the last column and row of P in (3), while for all other models $\boldsymbol{u} = \boldsymbol{v} = \boldsymbol{e}$). The matrix P is obtained normalizing by row $\widehat{M}$, that is $P = \mathrm{diag}(\widehat{\mathrm{M}}\,\boldsymbol{e})^{-1}\,\widehat{\mathrm{M}}$. Let

$$D = \mathrm{diag}(\widehat{\mathrm{M}}\,\boldsymbol{e})^{-1} = \begin{bmatrix} D(\boldsymbol{u}) & \\ & 1/(\boldsymbol{v}^T\boldsymbol{e}) \end{bmatrix},$$

where $D(\boldsymbol{u}) = \mathrm{diag}(\mathrm{M}\,\boldsymbol{e} + \boldsymbol{u})^{-1}$. Setting $\boldsymbol{x}^T = (\bar{\boldsymbol{x}}^T, x_{n+1})$, where $\bar{\boldsymbol{x}}^T$ is an n-vector, the equation $\boldsymbol{x}^T = \boldsymbol{x}^T\,P$ can be rewritten as

$$\begin{cases} \bar{\boldsymbol{x}}^T = \bar{\boldsymbol{x}}^T\,D(\boldsymbol{u})\,M + \dfrac{x_{n+1}}{\boldsymbol{v}^T\,\boldsymbol{e}}\,\boldsymbol{v}^T \\ \\ x_{n+1} = \bar{\boldsymbol{x}}^T D(\boldsymbol{u})\,\boldsymbol{u}. \end{cases} \tag{6}$$

Since we are interested in the direction of the Perron vector and not in its norm, we can chose $x_{n+1} = \boldsymbol{v}^T\,\boldsymbol{e}$, obtaining $\bar{\boldsymbol{x}}^T = \bar{\boldsymbol{x}}^T\,D(\boldsymbol{u})\,M + \boldsymbol{v}^T$. The vector $\bar{\boldsymbol{x}}$ is then the solution of the linear system

$$\Big(I - M^T D(\boldsymbol{u})\Big)\,\bar{\boldsymbol{x}} = \boldsymbol{v}, \tag{7}$$

or can be computed by the iterative method

$$\bar{\boldsymbol{x}}^{T\,(i+1)} = \bar{\boldsymbol{x}}^{T\,(i)} D(\boldsymbol{u})\,M + \boldsymbol{v}^T. \tag{8}$$

Note that for the `Stiff` models it is $D = I$.

It is important to observe that in the proposed models we can simply work with the matrices F_j without explicitly normalize and store the complete matrix $\widehat{M}$. For example, for the `Static` model in (4) the i-th block, $i = 1, \ldots, f$ of the vector $\widehat{M}\boldsymbol{e} = M\mathbf{e} + \mathbf{u}$, used for constructing matrix D, can be computed as follows

$$\mathbf{z}_i = \sum_{j \neq i} \alpha_{i,j}\,F_i^T F_j \mathbf{e}_j + \alpha_{i,i} F_i^T C F_i \mathbf{e}_i + \mathbf{e}_i, \quad i = 1, \ldots, f$$

and

$$\mathbf{z}_{f+1} = \sum_{j=1}^{f} \alpha_{f+1,j} F_j \mathbf{e}_j + \alpha_{f+1,f+1} \tilde{C}\,\mathbf{e}_{f+1} + \mathbf{e}_{f+1}.$$

The cost for computing $\mathbf{z}_i$ is linear in the number of non zeros (denoted as nnz) of all the matrices F_i and C, that is $O(\sum_i nnz(F_i) + nnz(C))$ since the matrices F_i are stored in a sparse format and the cost of multiplying a sparse matrix by a vector is equal to the number of non zeros in the matrix.

Letting $\boldsymbol{w}_i$ denote the vectors of length n_i, whose entries are the reciprocal of the entries of the vectors $\boldsymbol{z}_i$, and noticing that the i-th diagonal block of matrix $D(\mathbf{u})$ contains the entries of $\boldsymbol{w}_i$, an iteration of (6) becomes

$$\bar{\boldsymbol{x}}_j^{T\,(k+1)} = \begin{cases} \displaystyle\sum_{i \neq j} \alpha_{i,j} (\bar{\boldsymbol{x}}_i^{T\,(k)} * \mathbf{w}_i) F_i^T F_j + (\bar{\boldsymbol{x}}_j^{T\,(k)} * \mathbf{w}_j) F_j^T \tilde{C} F_j + \boldsymbol{e}_j \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \leq f \\ \\ \displaystyle\sum_{i=1}^{f} \alpha_{f+1,i} (\bar{\boldsymbol{x}}_i^{T\,(k)} * \mathbf{w}_j) F_i^T + \alpha_{f+1,f+1} (\bar{\boldsymbol{x}}_{f+1}^{T\,(k)} * \mathbf{w}_{f+1}) \tilde{C} + \boldsymbol{e}_{f+1} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = f+1, \end{cases}$$

where $*$ denotes the component-wise (Hadamard) product between vectors. The component-wise products can be computed in $O(\sum_i n_i + n_C)$ multiplications, and the total cost of computing the new vector $\bar{\boldsymbol{x}}^{T\,(k+1)}$ is proportional to the number of non zeros in the matrix $[F_1, F_2, \ldots F_f, C] + \sum_i n_i + n_C$. We can proceed analogously on the other models.

4. Solution of the Linear System with non-stationary methods

Once the problem of the computation of the Perron vector is reformulated as the solution of the linear system (7) we can employ the iterative method described in (8) or stationary methods such as Jacobi or Gauss-Seidel iterations, or the more promising Krylov methods. In fact, also non-stationary methods need only the computation of matrix-vectors products and are in general more effective than stationary ones (see [12, 15] for a comparison between stationary and non-stationary methods on similar problems). Recall that to compute the product $M\boldsymbol{x}$ then, we do not explicitly form and store the matrices M and $D(\boldsymbol{u})$ but we store in sparse form only the matrices of the features F_i and the citation matrix C.

We implemented different Krylov methods, and among them we chose the three more performing: BCGStab, CGS, TFQMR (see [23] for the details on these methods).

To refine the final result we add a few steps of the iterations (8) in accordance with the `Iterative Refinement` algorithm described below. In particular we perform some additional iterative step until either the distance of two successive iterations is less than `tol` or we are stuck and the vector is not changing anymore.

Procedure `Iterative Refinement`
Input: $\mathbf{x^{(i-1)}}, \mathbf{x^{(i)}}, \mathbf{x^{(i+1)}}$, `tol`
while $\|\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}\| <$ `tol` **or** $\left|\|\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)}\| - \|\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}\|\right| <$ `tol`
do a step of the iterative method (8), $i = i + 1$
endwhile

4.1. Models Validation: Stability and Convergence

To test the methods for the solution of (7) we constructed two datasets with real data extracted from the *US* patent office and we used five features: Firms, Inventors, Technologies, Lawyers and Examiners. In particular, we denote by F_1 the patent-technology matrix where entry (i, j) is one if patent i uses technology j; by F_2 the patent-firm matrix, recording the firm owning the patent, by F_3 the patent-inventors matrix that maps patents to inventors, by F_4 the patent-lawyers where each patent is matched to the lawyers applying for the patent, and by F_5 the matrix where at each patent is associated the examiners from the US Patent Office who approved the patent. The matrix C contains the citations between patents and is almost triangular since each patent can be based only on patents from the past.

DS1: Consists of $n_C = 2\,474\,786$ US patents from 1976-1990. Of these patents we have additional information that can be grouped into 5 major features, namely $n_1 = 472$ `Technologies`, $n_2 = 165\,662$ `Firms`, $n_3 = 965\,878$ `Inventors`, $n_4 = 25\,341$ `Lawyers` and $n_5 = 12\,817$ `Examiners`, giving rise to a matrix $\hat{A}$ of size $n_C + \sum_{i=1}^{5} n_i$ that is approximately of 3.7 millions.

models	BCGstab		CGS		TFQMR	
	it	$\log_{10}(res)$	it	$\log_{10}(res)$	it	$\log_{10}(res)$
Stiff-U	18	-10.49	100	-7.71	21	-3.90
Stiff-D	23	-11.77	100	-11.20	19	-4.75
Static-U	35	-9.03	100	-6.25	40	-7.83
Static-D	39	-11.13	100	-7.22	37	-9.60
Static-DD	35	-12.33	100	-12.20	30	-11.99
Heap-U	32	-9.86	100	-7.44	36	-8.59
Heap-D	36	-11.26	100	-7.73	38	-9.74
Heap-DD	41	-11.48	100	-9.46	33	-11.51
Heap-H	36	-10.83	100	-6.46	30	-7.72
Heap-HH	24	-9.85	100	-7.14	27	-8.38
SHeap-U	32	-11.56	100	-8.00	29	-9.91
SHeap-D	32	-11.72	100	-8.51	28	-9.85
SHeap-DD	37	-11.43	100	-11.83	28	-11.97
SHeap-H	28	-10.56	100	-8.33	25	-9.98
SHeap-HH	29	-11.34	100	-6.75	24	-10.05

Table 2: Performance comparison between three Krylov methods on the 15 models on a problem of size 3.7 million.

DS2: Consists of 7 984 635 US patents from 1976-2012. The size of the five features are as follows 475 `Technologies`, 633 551 `Firms`, 4 088 585 `Inventors`, 120 668 `Lawyers` and 64 088 `Examiners`, giving rise to a matrix $\hat{A}$ of size approximately of 13 millions.

The feature matrices and the citation matrix C are used to obtain ranks both for patents and features, i.e. Technologies, Firms, Inventors Lawyers and Examiners with the techniques described in this section.

When using iterative solvers we have always to address the question of numerical stability. The three proposed methods, `BCGStab`, `CGS` and `TFQMR` have been tested on the two datasets with an error goal of 10^{-11} and with maximum number of iterations equal to 100. For the refinement steps of the power method we set `tol` = 10^{-13}. Applying to dataset DS1 the three methods to all the models we obtain the results summarized in Table 2, where instead of the actual residuals we report only their base 10 logarithm.

It is evident that `CGS` is inadequate to cope with this kind of problems since after 100 iterations we have still a high residual norm. Moreover `BCGstab` is better then `TFQMR` since it achieves almost always a lower residual norm. For these reasons we restrict our analysis to `BCGStab` and `TFQMR` comparing them on the dataset of size 13M. We obtain the results reported in Table 3.

We note that `BCGstab` is clearly better than `TFQMR`, but sometimes fails to reach an acceptable accuracy. Hence a three step algorithm, described in Procedure `SystemSolver` has been devised.

models	BCGstab		TFQMR	
	it	$\log_{10}(res)$	it	$\log_{10}(res)$
Stiff-U	14	-10.57	19	-3.83
Stiff-D	21	-11.30	25	-3.71
Static-U	37	-6.77	52	-3.09
Static-D	52	-10.53	53	-7.89
Static-DD	39	-11.38	43	-8.70
Heap-U	36	-8.87	47	-7.58
Heap-D	45	-6.47	51	-6.11
Heap-DD	41	-9.40	41	-6.56
Heap-H	40	-9.63	43	-7.31
Heap-HH	35	-9.49	43	-7.52
SHeap-U	40	-9.78	38	-7.48
SHeap-D	38	-10.36	36	-8.10
SHeap-DD	36	-11.75	34	-9.79
SHeap-H	31	-7.90	35	-4.54
SHeap-HH	35	-10.63	35	-5.91

Table 3: Performance comparison between two Krylov methods applied to the 15 models of Table 1 on a problem of size 13 million.

Procedure `SystemSolver`
Input: Initial guess $\mathbf{x}^{(0)}$, `ErrorGoal`, `maxiter`, `tol`
Apply `BCGStab` with error goal=`ErrorGoal` and maximum iterations=`maxiter`
if `res` > `ErrorGoal`
  Apply `TFQMR` with error goal=`ErrorGoal` and maximum iterations=`maxiter`
endif
Apply `Iterative Refinement` with tolerance `tol`

Applying this procedure, with `ErrorGoal`=10^{-10}, `maxiter`=100 and `tol`=10^{-13}, on both the datasets we get the results displayed in Table 4.

From Table 4 we observe that the models that are more stable for the two datasets considered are the `Stiff-D`, `Static-DD`, and among the Heap-like models, we have good performance of `Heap-DD`, `SHeap-DD`.

5. Numerical Experiments

The problem of validating a ranking model is rather a difficult task since no ground truth is known in the general case. Moreover the validity of a model clearly depends on what we would like to measure. For example, if we want to measure the aptitude of a scholar to work in a team we will highly value the articles written in collaboration while if we want to measure the scientific strength and personal skills, we may want to normalize each of the articles by the number of co-authors. In this respect the extreme variety of our models and the different weighting strategies allows to tune the parameters to better satisfy the different needs.

models	DS1	size=3.7M	DS2	size =13M
	time(sec.)	$\log_{10}(res)$	time(sec.)	$\log_{10}(res)$
Stiff-U	237	-12.095	1078	-11.952
Stiff-D	179	-12.762	1422	-12.1884
Static-U	(*)239	-10.536	(*)2314	-9.0309
Static-D	188	-11.740	2096	-10.536
Static-DD	161	-13.002	1688	-11.7138
Heap-U	509	-11.138	(*)5992	-9.7579
Heap-D	467	-11.740	(*)7509	-10.536
Heap-DD	450	-12.535	(*)5849	-11.6019
Heap-H	440	-11.138	(*)5978	-9.93399
Heap-HH	(*)403	-11.439	(*)4999	-10.235
SHeap-U	80	-11.740	(*)717	-11.1381
SHeap-D	70	-11.439	661	-11.4391
SHeap-DD	67	-13.107	604	-12.3703
SHeap-H	86	-12.041	(*)662	-10.8371
SHeap-HH	60	-11.689	595	-10.536

Table 4: Performance of procedure SystemSolver on the 15 models on DS1 and DS2. The results labeled with $(*)$ are those where TFQMR has been applied since the required precision of 10^{-11} on the residual norm was not satisfied after 100 steps of BCGStab.

Table 5 summarize the experiments we performed on the two patents datasets. In the first set of experiments we compare the different ranking scores obtained with our models with simpler ranking methods, namely the Pagerank algorithm applied only to the citation matrix C, the ranking provided by one-class model and the simple citation count. The evaluation measure $P@N$ is also presented for comparing the top N ranked items by some of our models with simple citation count and PageRank. The top N firms obtains with some of our ranking methods are compared with the rank induced by number of patents issued by each form.

A second set of tests aims at showing that our different models are adequate to deal with incomplete data. In order to empirically prove that, we remove increasingly percentages of the attributes links to show that when dealing with incomplete database, our methods are still robust in providing a ranking "similar" to the one obtained with the full data. Of course, when the majority of the links are removed the rank should converge to the rank obtained with the One-class model. A direct comparison between the top ranked results with full and partial data is done as well.

With the third set of experiments we compare the ranking scores of the same algorithms with a finer or coarser aggregation in subclasses.

5.1. Comparison between models

The experiments reported in this section have different purposes. First we compare the rank provided by each model with the rank obtained with the one-class model, with the standard PageRank model and with the simple in-link counting. The idea is that the provided rank should differ substantially from the ranking obtained by simply counting the number of citations received, but the presence of the features should refine the ranking without completely reversing the importance of the players obtained by the one-class model or by the PageRank model.

Purpose	experiment	models	Section
Comparison (pat. and firms)	One-class	All	5.1
	PageRank	All	
	# Citations	All	
	P@N	StiffD, StaticDD,HeapHH, SHeapHH	
Incomplete data	$p = 0.1$	All	5.2
	$p = 0.5$	All	
	Top N vs p	StaticDD	
Consistence for class aggregation	finer	All	5.3
	coarser	All	

Table 5: Description of the experiments performed.

In Figure 2 it is shown the rank provided by our one-class model versus the rank provided by the standard PageRank algorithm [8]. A dot with coordinates (x_i, y_i) represents the i-th patent and x_i is the ranking score computed with the classical PageRank algorithm, and y_i the ranking score computed using our One-class model. We see that the two ranks are very alike because most of the points are located on a narrow strip along the main diagonal, reflecting the high correlation between the two ranks. In fact the only difference in the two models is the probability of reaching the dummy node that is 0.15 in the PageRank while it changes accordingly with the outdegree of each node in our model.

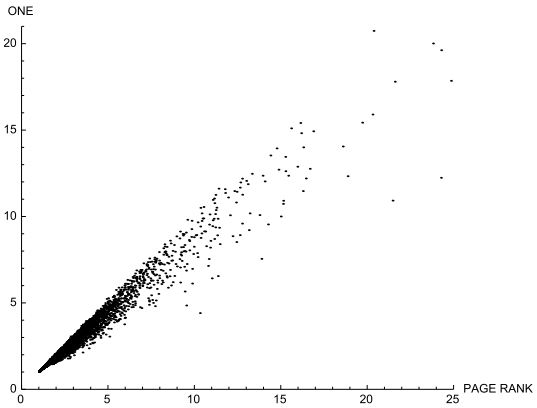

Figure 2: Comparison of the rank provided by the PageRank algorithm with random jump probability equal to 0.15 and the one obtained by the `one-class` model applied to DS1.

Examining the plots of the ranks obtained with all the models in Table 1 versus the number of citations received it turns out that the Uniform weighting scheme is not very adequate. In fact, for example in Figure 3(a), we see that there are objects that rank very high and have very few citations while some of those with many citations receive a very low rank value. This effect is less noticeable in the `Static` or `Heap` models but still the influence of number of citations on the actual ranking seems to be too weak. These problems together with the instability observed in previous section (see Tables 2, 3, 4 noting that for each model procedure `SystemSolver` performs better with other weighting schemes) suggest that uniform weighting strategies are inadequate.

The results provided by most of the models using a dimension based weighting scheme ap-

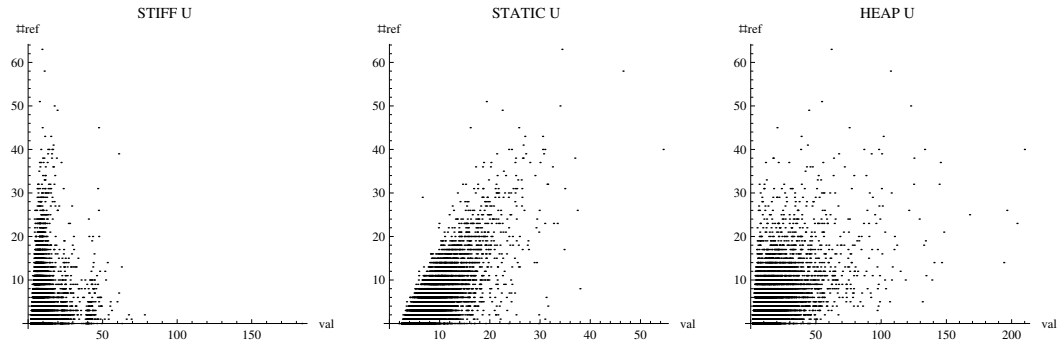

Figure 3: Comparison between the rank provided by three models with an Uniform weighting strategy (`val`) and citation count (`#ref`).

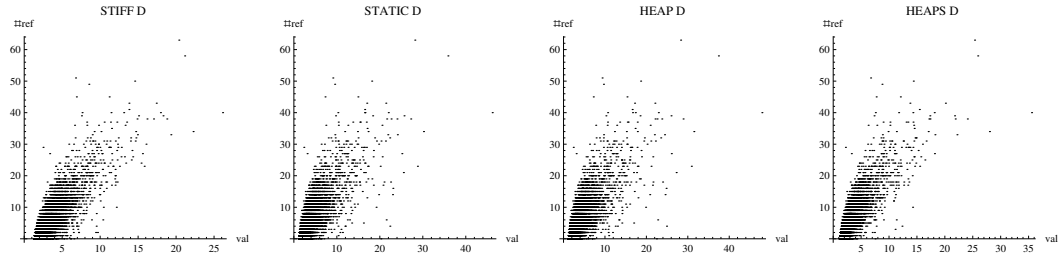

Figure 4: Comparison between the rank provided by three models with a dimensional based weighting strategy (`val`) and citation count (`#ref`).

pear to be better. In fact, documents with a high number of citations receive a good ranking score but the rank provided is not simply a citation count. As we can observe in Figure 4 there is not a substantial difference in the shape of the cloud of points obtained using different models. Similar results can be observed with double-dimension or heap weighting strategy.

Many authors use the *precision-at-N* ($P@N$) measure as evaluation method. This measure is defined as follows, for a given $N \in \mathbb{N}$

$$P@N = \frac{|E_N \cap F_N|}{N},$$

where E_N are the top ranked N objects according to the ranking method one has to evaluate, and F_N are the top ranked N objects accordingly with the "perfect" ranking. Of course since the "perfect" ranking is not available, the top objects are generally manually ranked by volunteers or other algorithms are taken into consideration. In our case, when ranking patents it is very hard to find reliable volunteers because of the expertise required to find the most valuable patents into a such large database. We used instead as comparison the rank provided by PageRank and the citation count. Figure 5 for values of $N = 50, 100, 200$ depicts the performance of four of our models, i.e. `Stiff-D`, `Static-DD`, `Heap-HH`, `SHeap-HH` respect to citation count (thick bars) and PageRank (thin bars). We note that our methods are more related to the rank produced by PageRank than to the simple citation count. The similarity is higher for the `SHeap` model since in that case the attributes are used in a less significant way. Surprisingly enough the `Static-DD`

model shares more of 60% of the top hits with PageRank, despite the two models are very different.

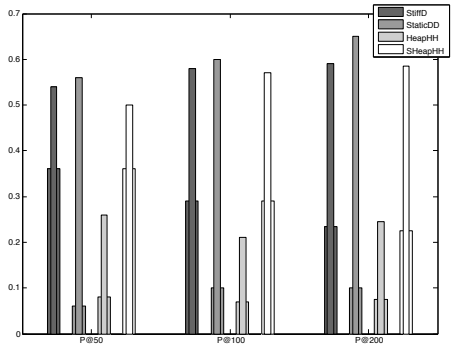

Figure 5: P@N performance of four of our models, i.e. Stif-D, Static-DD, Heap-HH, SHeap-HH respect to citation count (for each color the thick bars) and PageRank (for each color the thin bars)

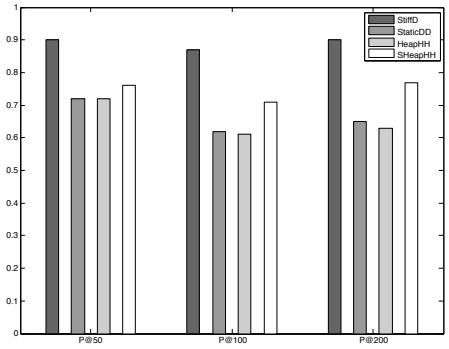

Figure 6: For firms the P@N performance of four of our models, i.e. Stiff-D, Static-DD, Heap-HH, SHeap-HH respect to number of patent granted to each firm shows an hight correlation.

The precision measure $P@N$ can be used also to evaluate the firms. In Figure 6 we show the comparison with the rank induced by sorting the firms by the number of patents issued. We see that there is a very high correlation with the number of patents issued by a given firm, up to 90% for the Stiff-D model. The precision is lower for the Heap-HH model where the citations matrix is combined with those of the features mitigating the effect of the the number of patents granted by a firm. In all the models in the top position we find very popular firms such as: IBM, Canon, Motorola, Philips, Sony, Bell *etc.*. Among the top results we have also firms such as Bell Labs, or Bayer AG, that despite in the time range [1976-2012] have issued a relatively low number of patents (2,617 and 896 respectively) show at the top of the list.

5.2. Convergence with incomplete data

An important problem when dealing with large collections of multivariate data is the incompleteness of the data. To see how robust our methods are when part of the data are missing, we performed many experiments leaving the citation matrix unaltered and varying the level of information about the features. In particular, we construct feature matrices $\tilde{F}_s$ obtained taking a nonzero from F_s with a fixed probability p, that is

$$P\left(\tilde{F}_s(i,j) = 1\right) = p\, F_s(i,j).$$

Then we replace in all the models the matrices F_s, $s = 1, \ldots, f$ with the matrices $\tilde{F}_s$.

The experiments performed have two different purposes. First, we would like to test if there are models for which the rank obtained decreasing the number of nonzero in the feature matrices does not converge to the one obtained with the one-class model. In fact a good model should exhibit a smooth convergence to the one-class model as p goes to zero. Second, we are interested to see if some of the models are predictive, in the sense that the rank obtained with missing data is "close enough" to the rank obtained using the full data, suggesting a good behavior when the data are partial or missing.

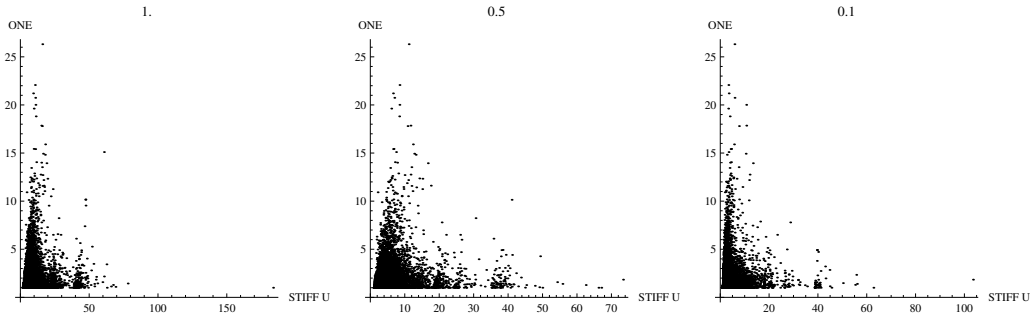

Figure 7: Comparison between the rank provided by the one-class model and the `Stiff-U` model for different values of the probability p.

We report some plots obtained for values of p equal to 1, 0.5 and 0.1. For $p = 0.1$ only 10% of the attributes are present so the ranking obtained should be very similar to the one obtained using only citations. Plotting the ranking values versus the rank obtained with the one-class model, we see that Uniform weighting schemas behave very poorly, since there is no convergence (see Figure 7). This fact, confirms the observation in the previous section about the inadequateness of Uniforms weighting strategies. On the contrary with the other weighting schemas all models exhibit a good convergence, showing the robustness to missing data. In Figure 8 and 9 are depicted the results for the three values of p for the `Static-DD` and the `Heap-H` models.

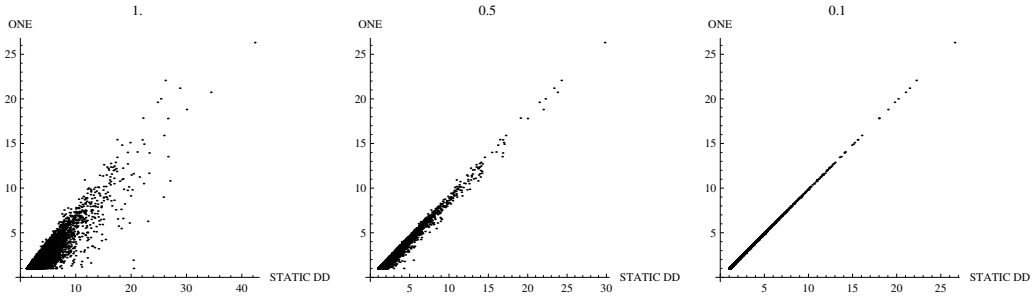

Figure 8: Comparison between the rank provided by the one-class model and the `Static-DD` model for different values of the probability p.

To better understand the effectiveness of the proposed methods when links are missing, we can compare the rank provided with all the links with that obtained using a small percentage of the link of the features. In Figure 10 and 11 are depicted the comparison between the rank of the patents for dataset DS1, and the rank of the patents using only 10% or 50% of the links of the features for the `Heap-H` model. We see that the rank obtained with partial information are not the same of those provided using the full matrix, but however the cloud has a reasonable shape, showing a good predictive properties of these models for missing data. Moreover, for lower percentage of missing links, the cloud is located in an thinner region around the diagonal.

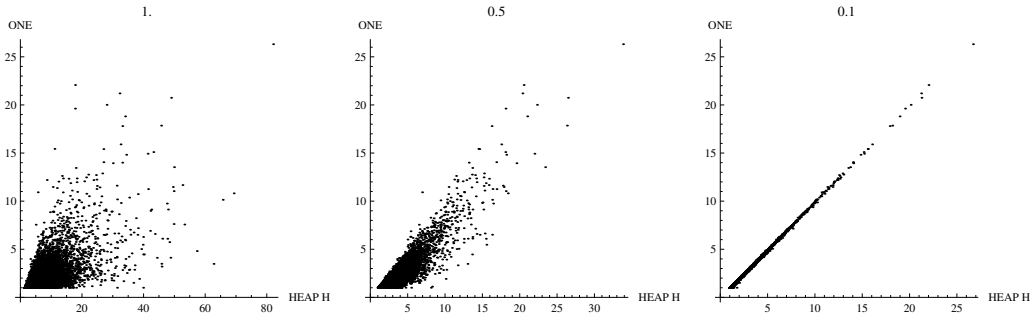

Figure 9: Comparison between the rank provided by the one-class model and the `Heap-H` model for different values of the probability p.

N	p=0.1	p=0.5
50	62%	66%
100	74%	77%
200	73%	77%

Table 6: Measure of intersection between the top N patents ranked using the `Static-DD` model and the rank obtained with the same model removing each edge of the attributes with probability 0.1 or 0.5).

For `Static-DD` model, and for the first N position in the ranked list, we measure the intersection between the rank provided with the full data and the one obtained with only 10% or 50% of the links of the attributes. The results in Table 6 show that the rank of the patents are very similar since among the top 100 patents we have that 77 are still in the top position even removing 50% of the links of the features, meaning that the most interesting patents show in the top position also with incomplete data.

5.3. Consistence for class aggregation

For some problems it is possible to tune the granularity of the subdivision in classes. For example, in our databases of patents we can decide how to group the technologies (in classes or subclasses) or geographical areas (regions or nations) and for scientific publications we can classify papers on the basis of their specific subject classification (there are many subject classifications tables such as AMS, MSC, ACM) or use a coarser grain based on disciplines. The granularity chosen depends of course on what the ranking is used for, but a good ranking schema should provide compatible results when using different granularities.

As an example, consider the patents in Table 7. All these patents are in the same class 15 (BRUSHING, SCRUBBING, AND GENERAL CLEANING) but have a secondary subclass as well. The rank of the patents obtained using the extended Technologies–Patent matrix should be similar to that obtained using a more compact Technologies–Patent matrix where, for example, the four patents in Table 7 are all grouped under the same Technology 15.

Figure 12 shows the comparison of the patents' ranks obtained using two different Technology-Patent matrices. In the compacted model we use only the main technology class, i.e. in the example of Table 7 the four patents associated with different subclasses will be classified as belonging to the same class 15. In the extended model, on the contrary, we will use a fatter Technology-Patent matrix, with a row for each different subclass. We see that the rank of the

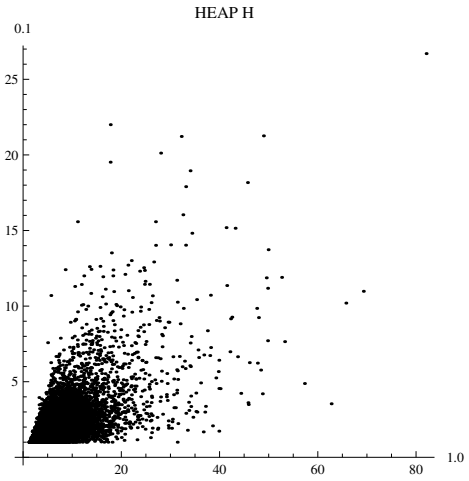

Figure 10: Comparison between the rank provided by the Heap-H model for the patents and the same model using only 10% of the links.

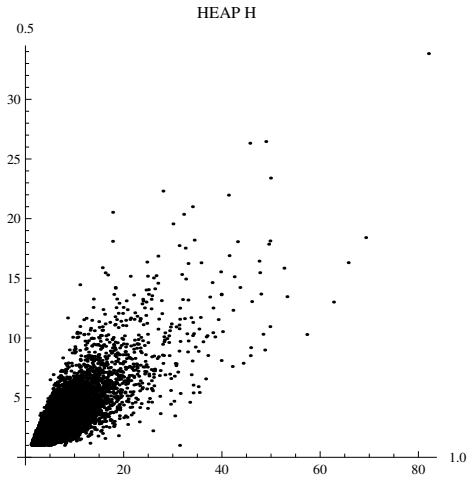

Figure 11: Comparison between the rank provided by the Heap-H model for the patents and the same model using only 50% of the links.

Patent number	Technology	Subclass
6895624	15	111 Brush and scraper
6895625	15	28 Rotary disk
6895626	15	50.1 Scrubber
6895627	15	98 Floor and wall cleaner:

Table 7: Four patents in the class 15- BRUSHING, SCRUBBING, AND GENERAL CLEANING, with different subclasses.

patents is minimally affected by the change. Of course the rank of Technologies changes a bit more. To compare the rank of the main 472 technologies (compact model) we summed up the rank of all the subclasses (extended model) of a technology.

The plot obtained using models with a weighting scheme of type DD are less grouped around the diagonal, meaning that the ranks obtained with the compact or extended technologies differ more than using the dimension based technique.

5.4. Considerations about the execution time

Our ranking algorithm works offline, in the sense that the scores are precomputed and stored as is done in Google's ranking algorithm. The computation of the rank can be done periodically. For example, for patents it is reasonable to update the rank weekly, while for scientific papers a recompilation after a month would be sufficient since most of the journals have monthly issues. Our algorithms require a time ranging from 20 minutes to 2 hours to compute the ranking on the larger dataset DS2 (where the matrix involved has size approximately of 13 millions) on a quad-core Intel Xeon @2.8GHz. To search among the documents one has to add a search module and retrieve the documents relevant to a given query. The ranking score of the relevant documents can be simply obtained pulling out from the list of all the documents sorted by ranking score.

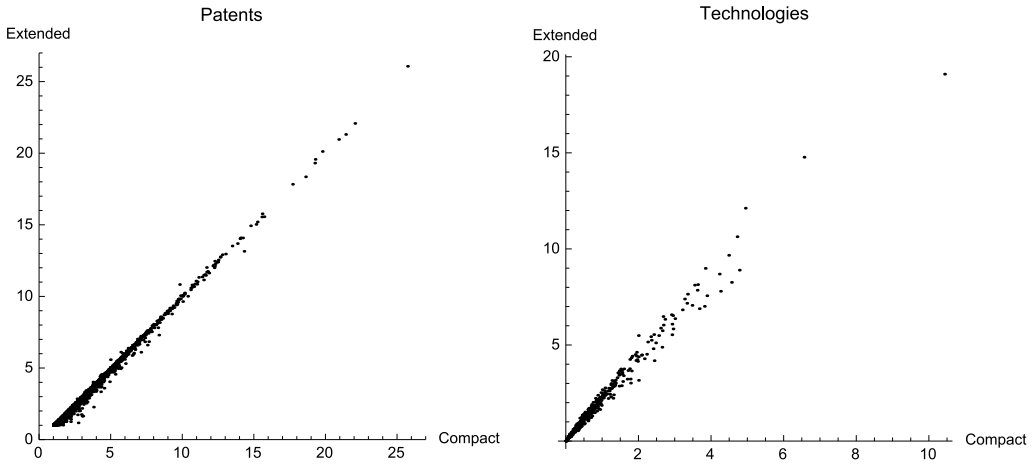

Figure 12: In the first picture a comparison between the rank of the patents provided by the `Static-D` model using the extended and compact Technology–Patent matrix. In the second picture the comparison of the ranks of Technologies for the extended and compact model.

6. Conclusion

In this paper we propose several models for ranking multi-parameters data on the basis of the linkage structure. We assume the citation matrix is enriched with other attributes (features) that can be represented by multi-class models. We use the attributes to improve the ranking process and, as a byproduct, we obtain a ranking of the attributes as well. After describing the models and different weighting strategies for measuring the influence of each feature in the ranking, we describe an algorithm for computing the rank based on an iterative scheme that combines non-stationary and stationary methods. We test some of the numerical methods on two large datasets of US patents. We address issues such as stability and convergence of the algorithm applied to each model, convergence with incomplete data, and consistence for class aggregation. In particular, the experimental part on large datasets shows that these techniques can be used in real applications where we have objects with multiple attributes and where some information can be missing due to the errors or incompleteness of the data. To search among the ordered list of objects one has to add a searching module and retrieve only objects relevant to a given query in analogy to what is done in the context of web search engines. A limitation of our approach is that the rank provided is static, hence the addition of a new document will require the computation of new ranks from scratch. We plan to investigate more efficient techniques for updating the ranks.

We also plan to address the problem of spam introducing mechanisms for penalizing self-citations and spammers. A possible approach to deal with cheating could consists in appropriately weighting citations and in modifying the main diagonal of the diagonal blocks to mitigate the influence of spammers on the final rank.

Another challenging future work is the incorporation of a preprocessing phase aimed at recovering missing entries. Unfortunately automatic techniques such the one proposed in [16] do not seem straightforwardly applicable to our case. However, a recovery strategy based on data

similarity could be successfully employed in specific domains. For example for bibliographic ranking one could use static indicators such as Impact Factor or Mathematical citation quotient. We plan to investigate how this information can be used in our scheme for improving the ranking process or as a starting point to reduce the number of iterations.

Acknowledgments

We would like to thank Monte J. Shaffer for the many discussions and for providing us the US patent office data. Many thanks also to the anonymous reviewers whose comments greatly improved the quality of the presentation.

This work was partially supported by GNCS-INDAM, and by the Research Project "Mathematical models and computational methods for complex networks" by the University of Pisa.

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, March 2011.

[2] P. D. Allison. Missing data. In Roger E. Millsap and Alberto Maydeu-Olivares, editors, *The SAGE: Handbook of Quantitative Methods in Psychology*, volume 3, pages 72–89. SAGE pub, 2009.

[3] K. Arrow. Economic welfare and the allocation of resources for invention. In *The Rate and Direction of Inventive Activity: Economic and Social Factors*, Universities-National Bureau, pages 609–626. Princeton University Press, 1962.

[4] G. Berardi, A. Esuli, F. Sebastiani, and F. Silvestri. Endorsements and rebuttals in blog distillation. *Information Sciences*, 249:38–47, 2013.

[5] D. A. Bini, G. M. Del Corso, and F. Romani. Evaluating scientific products by means of citation-based models: a first analysis and validation. *Electron. Trans. Numer. Anal.*, 33:1–16, 2008/2009.

[6] D. A. Bini, G. M. Del Corso, and F. Romani. A combined approach for evaluating papers, authors and scientific journals. *J. Comput. Appl. Math.*, 234:3104–3121, October 2010.

[7] E. Bozzo and D. Fasino. A multiparameter model for link analysis of citation graphs. *Electronic Transactions on Numerical Analysis*, 39:464–475, 2012.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[9] A.M. Buchanan and A.W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 316–322 vol. 2, 2005.

[10] F. Chung, P. Horn, and A. Tsiatas. Distributing antidote using pagerank vectors. *Internet Mathematics*, 6(2):237–254, 2009.

[11] W. M. Cohen, A. Goto, A. Nagata, R. R. Nelson, and J. P. Walsh. R&D spillovers, patents and the incentives to innovate in Japan and the United States. *Research Policy*, 31(8-9):1349–1367, 2002.

[12] G. M. Del Corso, A. Gullí, and F. Romani. Comparison of Krylov subspace methods on the PageRank problem. *Journal of Comput. and Appl. Math.*, 210:159–166, 2007.

[13] G. M. Del Corso and F. Romani. Versatile weighting strategies for a citation-based research evaluation model. *Bull. Belg. Math. Soc. Simon Stevin*, 16(4):723–743, 2009.

[14] D. M. Dunlavy, T. G. Kolda, and W. P. Kegelmeyer. Multilinear algebra for analyzing data with multiple linkages. Technical Report SAND2006-2079, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, April 2006.

[15] D. F. Gleich, L. Zhukov, and P. Berkhin. Fast parallel PageRank: A linear system approach. Technical Report YRL-2004-038, Yahoo! Research Labs, 2004.

[16] R. Guimerà and M. Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proceedings of the National Academy of Sciences*, 106(52):22073–22078, 2009.

[17] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[18] T. Kolda and B. Bader. The TOPHITS model for higher-order web link analysis. In *Proceedings of Link Analysis, Counterterrorism and Security 2006*, 2006.

[19] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, New York, NY, USA, 1987.

[20] Z. Nie, Y. Zhang, J. Wen, and W. Ma. Object-level ranking: bringing order to web objects. In *Proceedings of the 14th international conference on World Wide Web*, pages 567–574, New York, NY, USA, 2005.

[21] T. Okatani, T. Yoshida, and K. Deguchi. Efficient algorithm for low-rank matrix factorization with missing components and performance comparison of latest algorithms. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 842–849. IEEE, 2011.

[22] T. D. Pigott. A review of methods for missing data. *Educational research and Evaluation*, 7(4):353–383, 2001.

[23] Y. Saad. *Iterative methods for sparse linear systems (2nd Edition)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[24] J. L. Schafer. *Analysis of Incomplete Multivariate Data*. Chapman & Hall, London, 1997.

[25] J. Scheffer. Dealing with missing data. *Research Letters in the Information and Mathematical Sciences*, 3:153–160, 2002.

[26] M. Shaffer. *Entrepreneurial Innovation: Patent Rank and Marketing science*. PhD thesis, Washington State University, USA, 2011.

[27] J. Suchal. Enhancing search using layered graph ranking of multigraphs. In *IIT.SRC*, pages 1–8, 2008.

[28] Y. Sun and J. Han Mining heterogeneous information networks: A structural analysis approach. *SIGKDD Explorations*, 6(2):20–28, 2012.

[29] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 797–806, 2009.

[30] M. Trajtenberg. A penny for your quotes: Patent citations and the value of innovations. *Journal of Economics*, 21:172–1878, 1990.

[31] M-H. Tsai, C. Aggarwal, and T. Huang. Ranking in heterogeneous social media. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 613–622, New York, NY, USA, 2014. ACM.

[32] R. S. Varga. *Matrix iterative analysis*. Springer series in computational mathematics. Springer Verlag, Berlin, Heidelberg, Paris, 2000.

[33] C. Yu, R. Li, D. Yao, F. Lu, and H. Jin. Temporal-based ranking in heterogeneous networks. In C-H Hsu, X. Shi, and V. Salapura, editors, *Network and Parallel Computing*, volume 8707 of *Lecture Notes in Computer Science*, pages 23–34. Springer Berlin Heidelberg, 2014.

[34] Ming Zhang, Sheng Feng, Jian Tang, Bolanle Ojokoh, and Guojun Liu. Co-ranking multiple entities in a heterogeneous network: Integrating temporal factor and users bookmarks. In *Digital Libraries: For Cultural Heritage, Knowledge Dissemination, and Future Creation*, volume 7008 of *Lecture Notes in Computer Science*, pages 202–211. Springer Berlin Heidelberg, 2011.

[35] T. Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[36] T. Zhou, Linyuan Lü, and Yi-Cheng Zhang. Leaders in social networks, the delicious case. *PLoS ONE*, 6:623–630, 2011.

[37] X. Zhu, S. Zhang, Z. Jin, Z. Zhang, and Z. Xu. Missing value estimation for mixed-attribute data sets. *Knowledge and Data Engineering, IEEE Transactions on*, 23(1):110–121, 2011.