An Experimental investigation of kernels on graphs for collaborative recommendation and semi-supervised classification

François FOUSS, Université de Louvain à Mons
Kevin FRANCOISSE, Louvain School of Management
Luh YEN, Louvain School of Management
Alain PIROTTE, Louvain School of Management
Marco SAERENS, Louvain School of Management
An Experimental investigation of kernels on graphs for collaborative recommendation and semi-supervised classification

François Fouss, Université Catholique de Louvain à Mons
Kevin Françoisse, Louvain School of Management
Luh Yen, Louvain School of Management
Alain Pirotte, Louvain School of Management
Marco Saerens, Louvain School of Management

Summary

This paper presents a survey as well as an empirical comparison and evaluation of seven kernels on graphs and two related similarity matrices, that we globally refer to as “kernels on graphs” for simplicity. They are: the exponential diffusion kernel, the Laplacian exponential diffusion kernel, the von Neumann diffusion kernel, the regularized Laplacian kernel, the commute-time (or resistance-distance) kernel, the random-walk-with-restart similarity matrix, and finally, a kernel first introduced in this paper (the regularized commute-time kernel) and two kernels defined in some of our previous work and further investigated in this paper (the Markov diffusion kernel and the relative-entropy diffusion matrix). The kernel-on-graphs approach is simple and intuitive. It is illustrated by applying the nine kernels to a collaborative recommendation task, viewed as a link prediction problem, and to a semi-supervised classification task, both on several databases. The methods compute proximity measures between nodes that help study the structure of the graph. Our comparisons suggest that the regularized commute-time and the Markov diffusion kernels perform best on the investigated tasks, closely followed by the regularized Laplacian kernel.

Keywords: Kernels on graphs, Graph mining, Collaborative recommendation, Semi-Supervised classification.

Corresponding author:
François Fouss
Center for Excellence PRISME
Université Catholique de Louvain à Mons
Chaussée de Binche 151
B-7000 MONS, BELGIUM
Email: francois.fouss@uclouvain.be
An Experimental Investigation of Kernels on Graphs for Collaborative Recommendation and Semisupervised Classification

François Fouss\textsuperscript{a} Kevin Francoisse\textsuperscript{b} Luh Yen\textsuperscript{b} Alain Pirotte\textsuperscript{b} Marco Saerens\textsuperscript{b}

\textsuperscript{a}Management Sciences Department – LSM, Facultés Universitaires Catholiques de Mons (FUCaM), Belgium
\textsuperscript{b}Université catholique de Louvain, ICTEAM & LSM, Louvain-la-Neuve, Belgium

Abstract

This paper presents a survey as well as an empirical comparison and evaluation of seven kernels on graphs and two related similarity matrices, that we globally refer to as “kernels on graphs” for simplicity. They are: the exponential diffusion kernel, the Laplacian exponential diffusion kernel, the von Neumann diffusion kernel, the regularized Laplacian kernel, the commute-time (or resistance-distance) kernel, the random-walk-with-restart similarity matrix, and finally, a kernel first introduced in this paper (the regularized commute-time kernel) and two kernels defined in some of our previous work and further investigated in this paper (the Markov diffusion kernel and the relative-entropy diffusion matrix). The kernel-on-graphs approach is simple and intuitive. It is illustrated by applying the nine kernels to a collaborative-recommendation task, viewed as a link prediction problem, and to a semisupervised classification task, both on several databases. The methods compute proximity measures between nodes that help study the structure of the graph. Our comparisons suggest that the regularized commute-time and the Markov diffusion kernels perform best on the investigated tasks, closely followed by the regularized Laplacian kernel.

Key words: kernels on graphs, graph mining, collaborative recommendation, semisupervised classification.

1 Introduction

A function $k(\cdot, \cdot): \Omega \times \Omega \to \mathbb{R}$, for some input set or space $\Omega$ and real numbers $\mathbb{R}$, can be considered a similarity measure (i.e., enjoying natural properties of similarity like the triangular property (Chebotarev & Shamis, 1998a; Theodoridis & Koutroumbas, 2008)) if, given two objects $x$ and $y$ in $\Omega$, the real number...
k(x, y) characterizes in an intuitively adequate manner the similarities and differences (i.e., distance, proximity) of x and y. When the objects are expressed in a vector (or inner-product) space, one such simple and classical similarity measure is the inner product of x and y: k(x, y) = k(x, y) = x^T y.

Positive definite functions k(., .) – or kernel functions (Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004) – enjoy two important properties: (i) computing k(x, y) is equivalent to computing the inner product of some transformation of objects x and y into another, high-dimensional, space (often called the “feature space”, although this terminology is a bit confusing) where the data is more likely to be well-separated, while not requiring the explicit computation of the mapping from input space to feature space (only the pairwise similarities k(x, y) are required), and (ii) kernels allow to compute similarities between structured objects that cannot be naturally represented by a simple set of numbers. The latter property is illustrated in this paper, with the main objective of computing similarities between objects represented as nodes of a graph.

Thus, a useful kernel (Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004) is expected to capture an appropriate measure of similarity for a particular task (like classification or collaborative recommendation, as investigated in this paper) and to be computed efficiently directly from the original objects using a kernel function.

For each kernel function k(., .) and set of objects x_1, x_2, ..., x_n ∈ Ω, the n x n symmetric matrix K ([K]_{ij} = k_{ij} = k(x_i, x_j)) computing the kernel function for n x n pairs of objects is called a kernel matrix. It satisfies the following properties, that are all alternative definitions for symmetric positive semidefiniteness (see, e.g., (Meyer, 2000): (a) z^T K z ≥ 0 for all z ∈ R^n; (b) all the eigenvalues of K are non-negative; (c) K is a Gram, or inner product, matrix, and (d) K can be viewed as a diagonal matrix Λ ≥ 0 in another coordinate system (K = ΛU^T where U is an orthonormal matrix).

Various types of kernels are described in (Shawe-Taylor & Cristianini, 2004), namely polynomial kernels, Gaussian kernels, anova kernels, kernels on graphs, kernels on sets, kernels on real numbers, string kernels, and randomized kernels. This paper focuses on kernels on graphs: it gives an overview of the field, and it analyzes the results of applying various kernels to a collaborative-recommendation task, viewed as a link prediction problem, and to a semisupervised classification task. Kernels on graphs define similarity measures between graph nodes – the objects of interest. They should not be confused with so-called “graph kernels” that compute similarities between graphs (Vishwanathan, Schraudolph, Kondor, & Borgwardt, 2010; Gartner, 2008). As stated in (Vishwanathan et al., 2010), comparing nodes in a graph involves constructing a kernel between nodes, while comparing graphs involves constructing a kernel between graphs. In both cases, the research goal consists in defining kernels that capture the semantics inherent in the graph structure and that are reasonably efficient to evaluate.

Mining structured data, like graphs and relational databases, has been the focus of a growing interest (see for instance (Džeroski, 2003) and the other papers in that special issue on multi-relational data mining, or (Wilson, Han-
Kernels on graphs allow to address certain issues about graph structure. Indeed, a number of interesting measures, like a distance measure between graph nodes, are easily derived from a kernel. Distances between pairs of nodes allow, for example, to determine the items that are most similar to a given item, which facilitates labeling and clustering items. Intuitively-appealing measures of similarity between nodes are relatively easy to define. Still, their usefulness is clearly related to their ability to perform adequately in practice. This is precisely the objective of this paper: explore and evaluate nine such measures on real-world tasks of collaborative recommendation and classification.

The similarity measures studied in this paper integrate all paths (direct and indirect) between graph nodes. They have the nice property of increasing when the number of paths connecting two nodes increases and when the “length” of paths decreases. In short, the more short paths connect two nodes, the more similar those nodes are. On the contrary, the usual “shortest path” (also called “geodesic” or “Dijkstra” distance) between nodes of a graph does not necessarily decrease when connections between nodes are added and thus it does not capture the fact that strongly connected nodes are at a smaller distance than weakly connected ones. The similarity measures of this paper are usually easy to compute and have an intuitive interpretation, although they do not scale well for large graphs. Moreover, it is well-known that the distance measure naturally derived from a kernel matrix is Euclidean (see Section 2.2.4), that is, the nodes of the graph can be mapped to a Euclidean space preserving distances between nodes. This property can be used to, e.g., visualize the graph in a lower-dimensional space via a principal-component analysis (Saerens, Fouss, Yen, & Dupont, 2004; Scholkopf, Smola, & Muller, 1998; Shawe-Taylor & Cristianini, 2004) or a multidimensional scaling (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia, Kent, & Bibby, 1979) to define a subspace projection that enjoys some optimality property. More generally, kernel variants of multivariate statistical techniques can be used to explore the structure of the graph (Shawe-Taylor & Cristianini, 2004).

The contributions of this paper are threefold. (1) We introduce a new kernel on graphs called the regularized commute-time kernel and we further investigate two graph similarity matrices: the Markov diffusion kernel and the relative-entropy diffusion matrix (Fouss, Yen, Pirotte, & Saerens, 2006). (2) We provide a survey as well as (3) a comprehensive experimental comparison of those methods and of a number of kernels that were proposed in the literature, namely the exponential diffusion kernel (Kondor & Lafferty, 2002), the Laplacian exponential diffusion kernel (Kondor & Lafferty, 2002; Smola & Kondor, 2003), the von Neumann diffusion kernel (Shawe-Taylor & Cristianini, 2004), the regularized Laplacian kernel (Ito, Shimbo, Kudo, & Matsumoto, 2005; Smola & Kondor, 2003), the commute-time (or resistance-distance) kernel (Fouss, Pirotte, Renders, & Saerens, 2007; Saerens et al., 2004), and the random-walk-with-restart similarity matrix (Pan, Yang, Faloutsos, & Duygulu, 2006; Tong, Faloutsos, & Pan, 2006, 2008). A discussion of the rationale behind these kernels is also provided. The experiments indicate that three methods obtain good and stable performance, and that they outperform the other ones. Still, our objective here is not to develop state-of-the-art collaborative-recommendation or semi-supervised classification systems; rather, this paper aims at (i) providing a survey of the various graph similarity measures proposed in the literature, (ii) illustrat-
ing their usefulness for nontrivial database mining tasks, and (iii) comparing them.

We primarily investigate the extent to which the methods studied capture similarities between nodes. Thus the kernel-based scoring rules for both experimental tasks (recommendation and semisupervised classification) were directly based on simple averages or sums of these computed similarities. Of course, we expect that a finer tuning of similarities will improve the global results. The methods experimented here are, in fact, not kernel-based but rather similarity-based – that is, the methods do not require the similarity matrix to be a kernel matrix. Most of the similarity matrices tested are indeed positive semidefinite and therefore they can be regarded as kernel matrices, but that property is not exploited by the methods. For other state-of-the-art collaborative recommendation techniques, see for instance (Cheung, Tsui, & Liu, 2004; Delannay & Verleysen, 2008; Hofmann, 2004; Koren, 2008; Koren, Bell, & Volinsky, 2009; Paterek, 2007; Salakhutdinov, Mnih, & Hinton, 2007) and the references therein. The four last models were specifically developed in the context of the Netflix Prize. Concerning state-of-the-art semisupervised classification, the current trend is to design the kernel for the task at hand (see for instance (Johnson & Zhang, 2008; Kunegis & Lommatzsch, 2009; Lu, Jain, & Dhillon, 2009; Shaw & Jebara, 2009)); for a comprehensive survey of semisupervised classification, including graph-based classification, see (Abney, 2008; Chapelle, Scholkopf, & Zien, 2006; Zhu, 2008; Zhu & Goldberg, 2009).

This paper is an extended and improved follow-up of an earlier short conference paper (Fouss et al., 2006) presenting preliminary results. The main extensions in this paper concern (a) a more comprehensive literature review (Section 3), (b) the investigation of two additional algorithms (the random-walk-with-restart similarity matrix in Section 4.8 and the regularized commute-time kernel in Section 4.9) and of their link with the other kernels, (c) the introduction of the cosine method and the repositioning of the Basic method, previously called MaxF (Section 5.4), (d) a nested cross-validation (instead of a simple cross-validation) for assessing the performance of the algorithms (Section 5), (e) a thorough redesign of all experiments, and (f) the application of the algorithms to semisupervised classification (Section 6).

Besides this introduction, the paper is structured as follows. Section 2 reviews some relevant basic concepts from the theory of multidimensional scaling – borrowed from the support-vector-machine community (at least for finite-dimension spaces) – defining distance measures and optimal subspace projections from kernels on graphs (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979). Section 3 reviews related work on similarity measures and kernels on graphs. Section 4 briefly presents the nine kernels investigated. Sections 5 and 6 introduce the experiments that illustrate the kernel approach: the comparison between the nine kernels on two collaborative-recommendation tasks (Section 5) and on seven semisupervised classification tasks (Section 6). Section 7 provides a general discussion of the experimental results. Section 8 concludes the paper.

4
2 Theory review

This section briefly introduces the mathematical notations and the basic theory behind kernel matrices on graphs. In a very general setting, once some similarity measure between the nodes of a graph is shown to be a kernel, then a number of interesting properties automatically follow almost for free. More precisely, from a kernel matrix, it is possible to define a data matrix where each row represents a node of the graph (a node vector) as an \( r \)-dimensional vector in a Euclidean space (the feature space). These node vectors are such that their inner products are the elements of the kernel matrix. Through those node vectors, the graph can be viewed as a cloud of points in this \( r \)-dimensional space and standard multivariate statistical analysis technique become applicable.

2.1 Adjacency and Laplacian matrix of a graph

Consider a weighted, undirected, graph \( G \) with symmetric weights \( w_{ij} > 0 \) between pairs of nodes \( i \) and \( j \) linked by an edge. The elements \( a_{ij} \) of the adjacency matrix \( A \) of the graph are defined as usual as \( a_{ij} = w_{ij} \) if node \( i \) is linked to node \( j \) and \( a_{ij} = 0 \) otherwise. The Laplacian matrix is \( L = D - A \), where \( D = \text{Diag}(a_{i\cdot}) \) is the (generalized) degree matrix, with diagonal entries \( d_{ii} = \sum_{j=1}^{n} a_{ij} \), if the graph has \( n \) nodes in total. The volume of the graph is defined as \( V_G = \text{vol}(G) = \sum_{i=1}^{n} d_{ii} = \sum_{i,j=1}^{n} a_{ij} \). The weight \( w_{ij} \) of the edge connecting node \( i \) and node \( j \) can be set so that the more important the relation or affinity is between node \( i \) and node \( j \), then the larger the value of \( w_{ij} \) is. As for notations, the name of column vectors is written in bold lowercase while that of matrices is in bold uppercase.

If the graph is connected, that is, if every node is reachable from every node, then \( L \) has rank \( n - 1 \) (Chung, 1997). If \( e \) is a column vector all of whose elements are “1” (i.e., \( e = [1, 1, \ldots, 1]^T \), where \( T \) denotes the matrix transpose) and \( 0 \) is a column vector all of whose elements are “0”, \( Le = 0 \) and \( e^T L = 0^T \) hold; \( L \) is doubly centered. The null space of \( L \) is thus the one-dimensional space spanned by \( e \). Moreover, \( L \) is symmetric and positive semidefinite (see for instance (Chung, 1997)). Notice that, if the graph is not connected, it can be decomposed into closed subsets of nodes and the analysis can be applied independently on those subsets. Efficient algorithms for finding connected components are readily available (Jungnickel, 2008; Sedgewick, 1990).

2.2 Kernels on graphs

A real symmetric matrix that is not positive semidefinite can be changed to a positive semidefinite one by, e.g., adding the identity matrix \( I \) multiplied by a suitably positive constant to the original matrix (see, for example, (Roth, Laub, Buhmann, & Müller, 2003; Rousseuw & Molenberghs, 1993)). Indeed, if \( M \) is a symmetric matrix, the matrix \( (M + \alpha I) \), with \( \alpha \geq |\lambda_{\min}| \) (where \( \lambda_{\min} \) is the lowest eigenvalue of \( M \)) is positive semidefinite. This holds because \( M \) and \( (M + \alpha I) \) have the same eigenvectors \( u_i \) associated with eigenvalues \( \lambda_i \) for
\textbf{M} and \((\lambda_i + \alpha)\) for \((\text{M} + \alpha \text{I})\). Thus, by choosing \(\alpha \geq |\lambda_{\text{min}}|\), all the eigenvalues are non-negative and \((\text{M} + \alpha \text{I})\) is positive semidefinite.

Nine similarity matrices, defined from the adjacency matrix \(\text{A}\), are described in Section 4, and tested on real-world data in Sections 5 and 6. The main concepts basically come from the fields of multidimensional scaling (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979) and kernel methods (Gartner, 2008; Scholkopf & Smola, 2002; Shawe-Taylor & Cristianini, 2004).

\textbf{2.2.1 The data matrix associated to a kernel matrix}

Let us first recall a fundamental spectral-decomposition theorem from linear algebra, underlying kernel-based work (see, for example, (Mardia et al., 1979; Noble & Daniels, 1988)):

Any \(n \times n\) real positive semidefinite matrix \(\text{K}\) of rank \(r\) can be expressed as \(\text{K} = \text{U} \Lambda \text{U}^T = \text{U} \Lambda^{1/2} (\text{U} \Lambda^{1/2})^T = \text{XX}^T\), where \(\Lambda\) is the diagonal matrix containing the non-negative eigenvalues of \(\text{K}\), \(\text{U}\) is the \(n \times r\) orthonormal matrix whose columns are the normalized eigenvectors \(\text{u}_i\) of \(\text{K}\), and \(\text{X} = \text{U} \Lambda^{1/2}\) is a \(n \times r\) matrix of rank \(r\). Thus, the \(r\) columns \(\text{c}_i = \sqrt{\lambda_i} \text{u}_i\) of \(\text{X}\) correspond to the orthonormal eigenvectors \(\text{u}_i\) of \(\text{K}\) multiplied by the square root of their corresponding eigenvalue.

Therefore, if \(\text{x}_i = \text{row}_i(\text{X})^T\) is the column vector corresponding to column \(i\) of \(\text{X}^T\) (transposed row \(i\) of \(\text{X}\)), then the entries of \(\text{K}\) are inner products \([\text{K}]_{ij} = k_{ij} = \text{x}_i^T \text{x}_j\), in accordance with the fact that \(\text{K}\) is a kernel. The nodes of the graph are thus represented as vectors \(\text{x}_i\) in a \(r\)-dimensional Euclidean space. The \(\text{x}_i\)'s are called \textbf{node vectors}, while matrix \(\text{X} = [\text{x}_1, \text{x}_2, \ldots, \text{x}_n]^T\) containing the transposed node vectors as rows is the \textbf{data matrix} associated to the kernel matrix on a graph. This embedding is defined up to an isometry, but the coordinates of the node vectors obtained through spectral decomposition have the nice property of being expressed in the principal-component space (uncentered if the kernel is not centered).

The Euclidean space in which the node vectors are defined is the so-called feature space. Many techniques have been used to visualize the nodes in a low-dimensional space (Lee & Verleysen, 2007; Shawe-Taylor & Cristianini, 2004) but this question is not investigated in this paper.

\textbf{2.2.2 Centering the data matrix}

Usually, there is a preference for working with centered vectors, so that the centre of gravity of the node vectors \(\text{x}_i\) is \(\text{0}\). For instance, most multivariate statistical techniques assume that the data has been centered on the center of gravity of the data cloud (Jolliffe, 2002; Mardia et al., 1979). This amounts to subtracting its mean to each feature of the data matrix, therefore centering it. A data matrix is centered if \(\text{X}^T \text{e} = \text{0}\), that is, the sum of the elements of each column of \(\text{X}\) is 0. It can easily be shown that a symmetric kernel matrix \(\text{K}\) is centered (that is, it corresponds to inner products of centered node vectors) if and only if \(\text{Ke} = \text{0}\).
A centered kernel matrix $K$ is obtained by centering the node vectors, that is, by applying the symmetric centering matrix $H = I - n^{-1}ee^T$ (Mardia et al., 1979) to the data matrix $X$. Indeed, premultiplying a data matrix by $H$ re-expresses each element as a deviation from its column mean, i.e., $X = HX$ has its $(i,j)^{th}$ element $x_{ij} - \bar{x}_j$, where $\bar{x}_j$ is the mean of the $j^{th}$ column of $X$. Thus, the centered kernel matrix $\bar{K}$ is defined as $\bar{K} = (HX)(HX)^T = HXX^T H = HKH$.

### 2.2.3 The cosine, or normalized, kernel matrix

Inner products are not always an appropriate similarity measure. In some fields, such as information retrieval (Baeza-Yates & Ribeiro-Neto, 1999; Manning, Raghavan, & Schutze, 2008), the cosine similarity measure is preferred:

$$\cos_{ij} = \frac{x_i^T x_j}{||x_i|| ||x_j||} = \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}} \tag{1}$$

It corresponds to a kernel matrix since its elements are the inner products of the normalized node vectors $x_i / ||x_i||$ (Shawe-Taylor & Cristianini, 2004). In matrix form, $\text{Cos} = \text{Diag}(K)^{-1/2} K \text{Diag}(K)^{-1/2}$, where $\text{Diag}(K)$ is a diagonal matrix containing the diagonal of $K$.

### 2.2.4 Natural distance measure associated to a kernel matrix

A distance measure between pairs of nodes in the feature space (therefore also corresponding to a Euclidean distance between the node vectors of the data matrix) can be derived from the kernel matrix:

$$\delta_{ij}^2 = ||x_i - x_j||^2 = (x_i - x_j)^T(x_i - x_j) = x_i^T x_i + x_j^T x_j - 2x_i^T x_j = k_{ii} + k_{jj} - 2k_{ij} = (e_i - e_j)^T K (e_i - e_j) \tag{2}$$

where $e_i$ is a column vector with “0” entries, except in position $i$ whose entry is “1” (i.e., $e_i = [0, \ldots, 0, 1, 0, \ldots, 0]_n^T$).

Since $\delta_{ij}$ corresponds to a Euclidean distance in $\mathbb{R}^r$, it satisfies all the basic properties of a distance (positiveness, symmetry, and triangular inequality). Distances between pairs of elements allow, for instance, to use a clustering algorithm to group nodes that are most similar (Shawe-Taylor & Cristianini, 2004; Yen et al., 2005; Yen, Fouss, Decaestecker, Francq, & Saerens, 2007, 2008). In matrix form, $\Delta^{(2)} = \text{diag}(K)e^T + e(\text{diag}(K))^T - 2K$, where $\text{diag}(K)$ is a column vector containing the diagonal elements of $K$, and $[\Delta^{(2)}]_{ij} = \delta_{ij}^2$ contains the squared distances (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979).

With the cosine similarity measure, the distance reduces to $\delta_{ij} = \sqrt{2(1 - \cos_{ij})}$. The other way around, given a square Euclidean distance matrix $\Delta^{(2)}$, the natural centered kernel matrix associated to $\Delta^{(2)}$ is $K = -\frac{1}{2}H\Delta H$ (Borg & Groenen, 1997; Cox & Cox, 2001; Mardia et al., 1979).
Once a data matrix $X$ has been derived from $K$, standard multivariate statistical analysis methods can be used to investigate the structure of the data cloud. Kernel-based algorithms, such as kernel principal-component analysis, are applied directly to the kernel matrix without computing the data matrix in the feature space (Scholkopf et al., 1998; Shawe-Taylor & Cristianini, 2004).

3 Related work

This section provides an overview of the various similarity measures between nodes of a graph that were proposed in the literature, to the best of our knowledge, and that are investigated in this paper. Similarity between nodes has also been called relatedness (White & Smyth, 2003). The most well-known quantities measuring relatedness are co-citation (Small, 1973) and bibliographic coupling (Kessler, 1963). These are the two baseline measures that are adopted in our collaborative-recommendation experiments.

More sophisticated measures were also suggested. For instance, a similarity measure between nodes of a graph integrating indirect paths, based on the matrix-forest theorem, was proposed in (Chebotarev & Shamis, 1997, 1998b). While some nice properties were proven about this similarity measure, no experiment investigating its effectiveness was reported. It is in fact the same measure as the regularized Laplacian kernel described in Section 4.4.

A modified regularized Laplacian kernel was proposed in (Ito et al., 2005; Shimbo, Ito, Mochihashi, & Matsumoto, 2009) as an extension of the regularized Laplacian kernel, by introducing a new parameter controlling importance and relatedness. It was shown in (Ito et al., 2005; Shimbo & Ito, 2006; Shimbo et al., 2009) that the regularized Laplacian kernel overcomes some limitations of the von Neumann kernel (Kandola, Cristianini, & Shawe-Taylor, 2002), when ranking linked documents. This modified regularized Laplacian kernel is also closely related to a graph-regularization framework introduced by Zhou et al. in (Zhou, Bousquet, Lal, Weston, & Scholkopf, 2003) (see also (Belkin, Matveeva, & Niyogi, 2004; Wang, Wang, Zhang, Shen, & Quan, 2009; Yajima & Kuo, 2006; Zhou & Scholkopf, 2004)), and extended to directed graphs in (Zhou, Huang, & Scholkopf, 2005). The resulting kernel on graphs is a normalized version of the “regularized Laplacian kernel” involving the extension of the Laplacian matrix for directed graphs (Chung, 2005). In the same spirit, Chen et al. (Chen, Yang, & Tang, 2007) studied graph embedding and semisupervised classification by using this directed version of the regularized Laplacian kernel.

The exponential and the von Neumann diffusion kernels, based on the adjacency matrix, were introduced in (Kandola et al., 2002; Shawe-Taylor & Cristianini, 2004). They are computed through a power series of the adjacency matrix of the graph. They are therefore closely related to graph regularization models. Indeed, a general method for constructing natural families of kernels over discrete structures, based on the matrix-exponentiation idea, was introduced in (Kondor & Lafferty, 2002; Smola & Kondor, 2003) (see also (Lafferty & Lebanon, 2005). The focus of that work is on generating kernels on graphs, with the proposal of a special class of exponential kernels called diffusion kernels (see Sections 4.1 and 4.2). That work can be considered as the first attempt to explicitly define kernels on graphs.
The “commute-time” kernel was introduced in (Fouss et al., 2007; Saerens et al., 2004); it was inspired by the work of (Chandra, Raghavan, Ruzzo, Smolensky, & Tiwari, 1989; Klein & Randic, 1993). It takes its name from the average commute time, that is, the average number of steps that a random walker, starting from a given node, takes for entering another node for the first time and afterward returning to the starting node. The commute-time kernel is defined as the inner product in a Euclidean space where the nodes are exactly separated by the commute-time distance (proportional to the resistance-distance, see (Chandra et al., 1989; Klein & Randic, 1993)). This kernel performs well for collaborative recommendation, as documented in (Fouss et al., 2007). The same commute-time embedding was defined in (Qiu & Hancock, 2005, 2007; Ham, Lee, Mika, & Scholkopf, 2004; Yen et al., 2005) as well as (Brand, 2005), preserving the commute-time distance. It was applied to image segmentation and multi-body motion tracking (Qiu & Hancock, 2005, 2007), to dimensionality reduction of manifolds (Ham et al., 2004), to clustering (Yen et al., 2005), and to collaborative filtering (Brand, 2005; Fouss et al., 2007), with interesting results. The commute-time kernel is also closely related to the “Fiedler vector” (Fiedler, 1975b; Mohar, 1992), widely used for graph partitioning (Chan, Ciarlet, & Szeto, 1997; Pothen, Simon, & Liou, 1990) and clustering (Donetti & Munoz, 2004), as detailed in (Fouss et al., 2007), but also to discrete Green functions (Ding, Jin, Li, & Simon, 2007). Finally, a family of dissimilarity measures reducing to the shortest-path distance at one end and to the commute-time (or resistance) distance at the other end was proposed in (Yen, Mantrach, Shimbo, & Saerens, 2008) and subsequently in (Chebotarev, 2011) while a correlation or co-betweenness measure between nodes was developed in (Kolaczyk, 2009; Mantrach et al., 2009). The commute-time kernel is described in Section 4.5.

An intuitively appealing distance measure between nodes of a graph was proposed in (Latapy & Pons, 2005; Nadler, Lafon, Coifman, & Kevrekidis, 2005, 2006; Pons & Latapy, 2006) based on a continuous-time diffusion process, called the “diffusion distance”. We defined a discrete-time counterpart of their distance and used it to define the “Markov diffusion kernel” (see (Fouss et al., 2006) and Section 4.6 of this paper). An application of the diffusion distance to dimensionality reduction and graph visualization was described in (Lafon & Lee, 2006). The natural embedding induced by the diffusion distance was called “diffusion map” by Nadler et al. (Nadler et al., 2005, 2006). It is related to correspondence analysis (Yen, Saerens, & Fouss, 2011).

Other papers attempted to adapt the well-known PageRank procedure (Brin & Page, 1998; Page, Brin, Motwani, & Winograd, 1998) to define meaningful similarities between nodes. A random-walk with a restart procedure was proposed in (Pan et al., 2006; Tong et al., 2008). A random-walk process starting from a node of interest, controlled by a precomputed correlation matrix between nodes, was described in (Pucci, Gori, & Maggini, 2006). A random-walk-with-restart kernel is presented in Section 4.8.

A kernel approach for recommendation tasks based on one-class support-vector machines with kernels generated from a Laplacian matrix was proposed in (Yajima & Kuo, 2006). Their preliminary experiments on the MovieLens dataset produced interesting results with the commute-time kernel, the regularized commute-time kernel, and the regularized Laplacian kernel. For a
special choice of the parameters of the model, their technique reduces to computing a simple alignment of the kernel with a binary vector encoding the selected items of the active user. This scoring technique is in fact equivalent to a regularization technique used in semisupervised classification of graph nodes (Belkin et al., 2004; Johnson & Zhang, 2007, 2008; Wang et al., 2009; Zhou et al., 2003, 2005; Zhu, Ghahramani, & Lafferty, 2003) and it is the method of choice in our semisupervised classification experiments (see Section 6.2).

Several attempts to define similarity measures for directed graphs were proposed as well. For instance Chung (Chung, 2005) proposed to start from the transition-probability matrix $P = D^{-1}A$ where $D$ is, as before, a diagonal matrix containing the outdegrees of graph nodes. Then, they compute the left eigenvector $\pi$ of $P$, $\pi^T P = \lambda \pi^T$, which corresponds to the stationary distribution of the Markov chain. If the diagonal matrix containing the elements of $\pi$ on its diagonal is denoted by $\Pi = \text{Diag}(\pi)$, the Laplacian matrix of the directed graph is then defined by $L = \Pi - (\Pi P + P^T \Pi)/2$. It has a number of interesting properties generalizing those of the usual Laplacian matrix for undirected graphs. In particular, it is positive semidefinite and therefore it defines a kernel matrix. Still another extension of the Laplacian matrix to directed graphs has recently been proposed by Boley et al. (Boley, Ranjan, & Zhang, 2011). It corresponds to $L = D - A$. The authors show how the average commute times for a strongly connected directed graph are related to this asymmetric Laplacian matrix as a direct extension of similar well-known formulas for undirected graphs. Finally, yet another attempt to define similarities between nodes on a directed graph based on the matrix-forest theorem was described in (Agaev & Chebotarev, 2000, 2001).

The normalized Laplacian matrix can be defined as $\tilde{L} = \Pi^{-\frac{1}{2}} L \Pi^{-\frac{1}{2}}$. The regularized normalized Laplacian kernel matrix $(I - \alpha \tilde{L})^{-1}$ also appears in the context of semisupervised classification of labeled nodes of a graph (Zhou et al., 2005), while the regularized Laplacian kernel matrix is used for directed graph embedding in (Chen et al., 2007). Based on the same idea, Zhao et al. (Zhao & Tang, 2007) proposed a directed contextual distance and defined a directed graph from which the Laplacian matrix is computed. It was then used for ranking and clustering images. For undirected graphs, it can be shown that these kernel matrices reduce to variants of the regularized Laplacian kernel. Since all the graphs investigated in our experiments are undirected, we do not provide comparisons with these kernels.

4 Kernels on graphs

This section describes the nine kernels on graphs, already mentioned in Section 3. Several derived measures (a distance measure, the cosine normalized kernels, the PCA-based reduced-rank kernels) are also computed from these kernels and compared on the collaborative-recommendation task in Section 5.
4.1 The exponential diffusion kernel

The so-called exponential diffusion kernel $K_{ED}$, introduced in (Kondor & Lafferty, 2002), is defined as

$$K_{ED} = \sum_{k=0}^{\infty} \frac{\alpha^k A^k}{k!} = \exp(\alpha A) \quad (3)$$

where $A$ is the adjacency matrix of the graph and exp is the matrix exponential. Element $a_{ij}^k = [A^k]_{ij}$ of matrix $A^k$ ($A$ to the power $k$) is the number of paths between nodes $i$ and $j$ with exactly $k$ transitions or steps. Thus, the kernel integrates a contribution from all paths between node $i$ and node $j$, discounting paths according to their number of steps. It favors shorter paths between two nodes by giving them a heavier weight. The discounting factor is $\alpha^k/k!$. Other choices for that factor lead to other kernels, like the von Neumann diffusion kernel, described in Section 4.3.

The $K_{ED}$ matrix is clearly positive semidefinite since the exponential of $A$ amounts to replacing $\lambda$ (a diagonal matrix containing the eigenvalues of $A$) by $\exp(\lambda)$ in the spectral decomposition of $A$ (the matrix exponential of a diagonal matrix is a diagonal matrix whose $(i,i)$ element is the exponential of the corresponding element in the original matrix, therefore always positive).

4.2 The Laplacian exponential diffusion kernel

A meaningful alternative to $K_{ED}$ relies on a diffusion model (Smola & Kondor, 2003) that substitutes the adjacency matrix with minus the Laplacian matrix in Equation (3). It is closely related to continuous-time Markov chains (Kelly, 1979; Norris, 1997; Parzen, 1962). As an intuitive insight, suppose that a quantity $x_i(t)$ is defined on each node $i$ of the graph at time $t$ and that it diffuses to neighboring nodes with a symmetric diffusion rate $a_{ij}$. Thus, during a small time interval $\delta t$, an amount $a_{ij} x_i \delta t$ is transferred from node $i$ to node $j$, proportional to $\delta t$, $a_{ij}$, and $x_i(t)$. The balance equation (input and output during $\delta t$) is given by

$$x_i(t + \delta t) = x_i(t) + \sum_{j=1}^{n} a_{ji} x_j \delta t - \sum_{j=1}^{n} a_{ij} x_i \delta t \quad (4)$$

Since $A$ is symmetric, this leads to, for $\delta t \to 0$,

$$\frac{dx_i(t)}{dt} = \sum_{j=1}^{n} a_{ji} x_j - a_{ii} x_i = \sum_{j=1}^{n} (a_{ji} x_j - a_{ii} \delta_{ij} x_j)$$

$$= - \sum_{j=1}^{n} (a_{ii} \delta_{ij} - a_{ij}) x_j \quad (5)$$
where $\delta_{ij}$ is the Kronecker delta. The elements $l_{ij} = (a_i \delta_{ij} - a_{ij})$ are the entries of the Laplacian matrix. In matrix form,

$$\frac{dx(t)}{dt} = -Lx(t)$$

This system of differential equations leads to the following solution

$$x(t) = \exp(-Lt)x_0$$

where $x_0$ is the initial vector $x$ at time $t = 0$ and $\exp$ is the matrix exponential. This leads to the Laplacian exponential diffusion kernel $K_{LED}$, introduced in (Kondor & Lafferty, 2002; Smola & Kondor, 2003), and defined as

$$K_{LED} = \exp(-\alpha L)$$

which is similar to Equation (3), with minus the the Laplacian matrix replacing the adjacency matrix. If we substitute $x_0 = e_j$ in Equation (7), then the equation shows that column $j$ of $K_{LED}$ corresponds to quantity $x$ at time $t = \alpha$ when the initial vector is $e_j$, that is, a unit quantity starting diffusing from node $j$ at $t = 0$.

### 4.3 The von Neumann diffusion kernel

The von Neumann diffusion kernel $K_{VND}$ (Kandola et al., 2002; Shawe-Taylor & Cristianini, 2004) differs from the exponential diffusion kernel by its exponential discounting rate $\alpha^k$, instead of $\alpha^k/k!$:

$$K_{VND} = \sum_{k=0}^{\infty} \alpha^k A^k = (I - \alpha A)^{-1}$$

$K_{VND}$ is well defined if $0 < \alpha < \rho(A)^{-1}$ where $\rho(A)$ is the spectral radius of the symmetric matrix $A$ (that is, its largest eigenvalue in absolute value). In this case, $K_{VND}$ is positive definite. It was shown in (Ito et al., 2005) that this kernel applied to document ranking defines a link-analysis measure intermediate between co-citation/bibliographic coupling (Kessler, 1963; Small, 1973) and HITS (Kleinberg, 1999). Indeed, von Neumann kernels subsume Kleinberg’s HITS importance at the limit of their parameter range. Because they reduce to co-citation relatedness at the other end of the parameter, von Neumann kernels provide a spectrum of link analysis measures between the two established measures of importance and relatedness (Shimbo et al., 2009).

### 4.4 The regularized Laplacian kernel

The regularized Laplacian kernel (Chebotarev & Shamis, 1997, 1998b; Ito et al., 2005; Smola & Kondor, 2003) is also called the normalized random walk
with restart kernel in (Mantrach et al., 2011) because of its links with the random walk with restart model (Pan et al., 2006; Tong et al., 2006, 2008) (see Section 4.8. It differs from the von Neumann diffusion kernel only by substituting the negated Laplacian matrix $-L$ for the adjacency matrix $A$:

$$K_L = \sum_{k=0}^{\infty} \alpha^k (-L)^k = (I + \alpha L)^{-1}$$

where $0 < \alpha < \rho(L)^{-1}$ and $\rho(L)$ is the spectral radius of $L$. $K_L$ is clearly positive definite. The regularized Laplacian kernel has been shown to provide better performance than the von Neumann kernel in a bibliographic citation task (Ito et al., 2005).

This similarity measure has an interesting interpretation in terms of the matrix-forest theorem (Chebotarev & Shamis, 1997, 1998b). Let $F^i$ be the set of all spanning forests rooted at node $i$ of graph $G$ and $F^{ij}$ be the set of those spanning rooted forests for which nodes $i$ and $j$ belong to the same tree rooted at $i$. A spanning rooted forest is an acyclic subgraph of $G$ that has the same nodes as $G$ and one marked node (a root) in each component. It is shown in (Chebotarev & Shamis, 1997, 1998b) that the matrix $(I + L)^{-1}$ exists and that $[(I + L)^{-1}]_{ij} = \epsilon(F^{ij})/\epsilon(F^i)$ where $\epsilon(F^{ij})$ and $\epsilon(F^i)$ are the total weights of forests that belong to $F^{ij}$ and $F^i$ respectively. The elements of this matrix are therefore called “relative forest accessibilities” between nodes. This interpretation can be generalized to the matrix $(I + \alpha L)^{-1}$ where parameter $\alpha$ weights the number of edges belonging to the forests and limits the size of forests in terms of number of edges (see (Chebotarev & Shamis, 1997, 1998b) for more detail). This matrix can be shown to be a similarity measure.

This kernel has also been used for semisupervised classification in the following regularization framework (Belkin et al., 2004; Sindhwani, Belkin, & Niyogi, 2006; Wang et al., 2009; Zhou et al., 2003, 2005; Zhu et al., 2003). Let a class membership $y_c^i$ be associated to each node $i$ with value equal to 1 when node $i$ belongs to class $c$ and 0 otherwise. Label consistency is assumed, namely that neighbor nodes are likely to have the same class label (Kolaczyk, 2009; Zhou et al., 2003). This assumption is also usually made in spatial statistics (Cressie, 1993; Haining, 2003) and often called the “cluster assumption”, “guilt by association”, or “structural autocorrelation”.

A reasonable smoothed, predicted, class membership value, $\hat{y}_c^i$, can be obtained by minimizing

$$\sum_{i=1}^{n}(\hat{y}_c^i - y_c^i)^2 + \alpha \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(\hat{y}_c^i - \hat{y}_c^j)^2$$

Other supervised classification methods, such as k-nearest neighbors, also rely on such a label consistency assumption. This loss function consists of two parts: a penalty accounting for the deviation of the predicted value from the real value (quality of fit term) and a smoothness term imposing that nearby nodes have similar predicted values (such a smoothing term was already suggested in (Hall, 1970)). With $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(\hat{y}_c^i - \hat{y}_c^j)^2 = 2\hat{y}_c^T L \hat{y}_c$ (Hall, 1970), minimizing
this loss function in terms of the predicted values yields

\[ \hat{y}^c = (I + \alpha L)^{-1} y^c \]  

(12)

which aims to simply compute the sum of similarities with every node of the graph belonging to class \( c \) – an alignment with the binary vector encoding class labels\(^1\). This simple sum-of-similarities procedure is used in Section 6.2 for the semisupervised classification (SC) experiments. For other choices of the quadratic penalty and smoothing terms, minimization of the loss function results in the product of a graph similarity matrix, similar to the kernels on graphs discussed in this paper, with a membership vector \( y^c \) (Zhou et al., 2003). Interestingly, the same technique also appears in collaborative recommendation with one-class support vector machine, for a special choice of the parameters (Yajima & Kuo, 2006).

Finally, a modified regularized Laplacian kernel was proposed in (Ito et al., 2005) that introduces a parameter controlling importance and relatedness. A modified Laplacian matrix is defined as \( L_{\gamma} = \gamma D - A \) with \( 0 < \gamma \leq 1 \).

The **modified regularized Laplacian kernel** \( K_{RL} \) (Ito et al., 2005) is thus computed as

\[
K_{RL} = \sum_{k=0}^{\infty} \alpha^k (-L_\gamma)^k = (I + \alpha L_\gamma)^{-1}
\]

(13)

It was shown that, for \( \alpha > 0 \) and \( 0 < \gamma \leq 1 \), if the series (13) converges, \( K_{RL} \) is positive semidefinite and that it yields a measure intermediate between relatedness and importance (Ito et al., 2005). Since \( K_{RL} \) is more general than \( K_L \), it will be used in our experiments. Indeed, \( K_{RL} = K_L \) when \( \gamma = 1 \).

### 4.5 The commute-time kernel

Consider a discrete Markov chain defining a random walk on graph \( G \) and let \( s(t) \) be a random variable representing the state of the Markov chain at time step \( t \). Thus, if the process is in state \( i \in \{1, \ldots, n\} \) at time \( t \), then \( s(t) = i \). Let the probability of being in state \( i \) at time \( t \) be denoted as \( x_i(t) = P(s(t) = i) \) and define \( P \) as the transition-probability matrix with entries \( p_{ij} = [P]_{ij} = P(s(t + 1) = j | s(t) = i) \). Consider a Markov model whose transition probabilities are \( p_{ij} = a_{ij} / a_{i*} \) with \( a_{i*} = \sum_{j=1}^{n} a_{ij} \). In other words, to any state or node \( s(t) = i \), a probability of jumping to an adjacent node \( s(t+1) = j \) is associated that is proportional to the weight \( a_{ij} \geq 0 \) of the edge connecting \( i \) and \( j \), and then normalized (this corresponds to a standard random-walk model on a graph). Transition probabilities depend only on the current state and not on past ones (first-order Markov chain). Since the graph

\(^1\) Strictly speaking, in the context of semisupervised classification, this least square should be defined on the training set only (labeled samples). However, initially, the framework of Zhou et al. (Zhou et al., 2003) considers that, when facing unlabeled samples, the function should be close to 0 as well. In other words, unlabeled samples are considered as not belonging to the class of interest. See, e.g., (Bengio, Delalleau, & Le Roux, 2006) for the model taking unlabeled samples into account.
is undirected and connected, the Markov chain is reversible and irreducible (every state can be reached from every state). In matrix form, the evolution of the Markov chain is characterized by \( \mathbf{x}(t+1) = \mathbf{P}^T\mathbf{x}(t) \), which provides the state probability distribution \( \mathbf{x}(t) = [x_1(t), x_2(t), ..., x_n(t)]^T \) at time \( t \) given the initial probability density \( \mathbf{x}(0) = \mathbf{x}_0 \) at \( t = 0 \). After \( t \) steps, the probability distribution is given by \( \mathbf{x}(t) = (\mathbf{P}^T)^t\mathbf{x}_0 \).

The commute-time kernel (Fouss et al., 2007; Saerens et al., 2004) takes its name from the average commute time \( n(i, j) \), which is the average number of steps that a random walker, starting in node \( i \neq j \), takes before entering node \( j \) for the first time and then going back to \( i \). The average commute time (Klein & Randic, 1993; Saerens et al., 2004; Fouss et al., 2007) can be computed as

\[
n(i, j) = V_G (e_i - e_j)^T L^+ (e_i - e_j)
\]

where every node \( i \) of the graph is represented by a basis vector \( e_i \) in the Euclidean space \( \mathbb{R}^n \) and \( V_G \) is the volume of the graph. \( L^+ \) is the Moore-Penrose pseudoinverse of the Laplacian matrix of the graph. It is positive semidefinite. Thus, Equation (14) is a Mahalanobis distance between the nodes of the graph. It is referred to as the “commute-time distance” or the “resistance distance” because of a close analogy with the effective resistance in electrical networks (Chandra et al., 1989; Klein & Randic, 1993; Fouss et al., 2007). However, it has been shown that this resistance distance is in fact meaningless when dealing with large graphs (Luxburg, Radl, & Hein, 2010) so that it should be used with caution.

It can be shown that the elements of \( L^+ \) are inner products of the node vectors in the Euclidean space where these node vectors are exactly separated by commute-time distances (Brand, 2005; Saerens et al., 2004). In other words, the elements of \( L^+ \) can be viewed as similarity measures between nodes. Hence the **commute-time kernel** \( K_{CT} \) (Fouss et al., 2007; Saerens et al., 2004) is defined as

\[
K_{CT} = L^+
\]

with no parameter tuning necessary. The dominant eigenvector of \( K_{CT} \) is in fact the Fiedler vector (Fouss et al., 2007) known to convey some important information concerning the structure of the graph (Fiedler, 1975a, 1975b, 1989; Chung, 1997). A slight modification of this kernel, the sigmoid commute-time kernel, worked remarkably well in a node-clustering task (community detection, (Yen et al., 2007; Yen, Fouss, et al., 2008)).

Because of some issues with the \texttt{pinv} function in Matlab when dealing with large sparse matrices, the following formula computing the pseudoinverse of the Laplacian matrix (see, e.g., (Rao & Mitra, 1971), chapter 10) was used in all our experiments:

\[
L^+ = \left( L - ee^T/n \right)^{-1} + ee^T/n
\]

This formula exploits the fact that \( L \) is centered and thus \( e \), a column vector full of 1’s, spans its null space.

There is of course a close relationship between the commute-time kernel and the regularized Laplacian kernel. Indeed, if \( L \) has eigenvalues \( \lambda_i \) (in decreasing
order), $(L + \alpha^{-1}I)$ has corresponding eigenvalues $(\lambda_i + \alpha^{-1})$ and both matrices share the same eigenvectors $u_i$. Remember that $L$ and its Moore-Penrose pseudoinverse $L^+$ have the same set of eigenvectors and inverse eigenvalues (zero eigenvalues are equal to zero in both matrices). The Laplacian matrix of a connected graph has rank $n-1$ and therefore it has exactly one eigenvalue equal to 0 (Chung, 1997). Thus a spectral decomposition of the corresponding kernel yields $L^+ = \sum_{i=1}^{n-1} \lambda_i^{-1} u_i u_i^T$ (by definition of the Moore-Penrose pseudoinverse) and $(I + \alpha L)^{-1} = \alpha^{-1} \sum_{i=1}^n (\lambda_i + \alpha^{-1})^{-1} u_i u_i^T$. Since the normalized eigenvector of $L$ corresponding to eigenvalue $\lambda_n = 0$ is $u_n = e/\sqrt{n}$, the $n$th term for the commute-time kernel is simply 0, while it is $ee^T/n$ for the regularized Laplacian kernel. Therefore, this last term simply adds a constant value $(1/n)$ to all the elements of the matrix and, if $\alpha$ is large, $\lambda_i + \alpha^{-1}$ will be close to $\lambda_i$, so that the two kernels essentially differ by a multiplication factor and a constant value added to all the elements of the matrix.

4.6 The Markov diffusion kernel

This kernel introduced in (Fouss et al., 2006) is based on a diffusion distance defined in (Latapy & Pons, 2005; Nadler et al., 2005, 2006; Pons & Latapy, 2006) between nodes of a graph in a diffusion model. We adapted their definition of diffusion distance to periodic Markov chains (as nonperiodic Markov chains are unrealistic for collaborative recommendation) to define a valid kernel on a graph (Fouss et al., 2006; Yen et al., 2011). It was shown in (Nadler et al., 2005, 2006) that the low-dimension representation of the data by the first few eigenvectors of the corresponding transition-probability matrix is optimal under a given least-square error criterion involving the diffusion distance. The intuition behind the diffusion distance is the following: two nodes are considered similar when they diffuse in a similar way through the graph, and therefore when they “influence” the other nodes in a similar manner. An application of the diffusion distance to dimensionality reduction and graph visualization was described in (Lafon & Lee, 2006).

More precisely, the average visiting rate $\pi_{ik}(t)$ in state $k$ after $t$ steps (the probability of finding the random walker in state $k$ during the first $t$ steps) for a process that started in state $i$ at time $t = 0$ is computed as follows:

$$\pi_{ik}(t) = \frac{1}{\ell} \sum_{\tau=1}^{t} P(s(\tau) = k | s(0) = i)$$ (17)

From this quantity, the diffusion distance at time $t$ between node $i$ and node $j$ is defined as follows

$$d_{ij}(t) = \sum_{k=1}^{n} (\pi_{ik}(t) - \pi_{jk}(t))^2$$ (18)

which corresponds to the sum of the squared differences between the average visiting rates after $t$ transitions when starting from node $i$ and node $j$ at time $t = 0$. This is a natural definition which quantifies the dissimilarity between two nodes based on the evolution of the probability distribution. Thus, if
two nodes influence (diffuse through) the graph in exactly the same way, the
distance between them is zero. Of course, when \( i = j \), \( d_{ij}(t) = 0 \).

This diffusion distance was proposed by Nadler et al. (Nadler et al., 2005) and
by Latapy and Pons (Latapy & Pons, 2005) in the context of diffusion pro-
cesses. Actually, their original definition, namely
\[
d_{ij}(t) = \sum_{k=1}^{n} (x_{ik}(t) - x_{jk}(t))^2,
\]
i

involves \( x_{ik}(t) \), the probability of finding the random walker in state \( k \) at time \( t \),
when starting from node \( i \) at time 0 (i.e., \( x(0) = e_i \)) instead of \( \pi_{ik}(t) \) as
in Equation (18). Their definition does not work with periodic Markov chains
that are relevant to collaborative recommendation (with a bipartite graph).
The definition in (Latapy & Pons, 2005; Nadler et al., 2005, 2006; Pons &
Latapy, 2006) also adds a weighting factor proportional to the inverse of the
stationary distribution of the Markov chain, therefore putting more weight on
low-density nodes. We do not include that factor here because experiments
showed that it does not improve performance for the applications studied in
Sections 5 and 6. Still another variant of this diffusion kernel was recently
introduced in (Yen et al., 2011).

We first compute \( \pi_{ik}(t) \) from Equation (17):
\[
\pi_{ik}(t) = \frac{1}{t} \sum_{\tau=1}^{t} P(s(\tau) = k|s(0) = i) = \frac{1}{t} \sum_{\tau=1}^{t} e_k^T(P^T)^{\tau}e_i = e_k^T \left[ \frac{1}{t} \sum_{\tau=1}^{t} P^\tau \right]^T e_i
\]
since \( P(s(\tau) = k|s(0) = i) = e_k^T x(t) = e_k^T(P^T)^{\tau}e_i \) when \( x_0 = e_i \). By defining
\( Z(t) = \frac{1}{t} \sum_{\tau=1}^{t} P^\tau \), we obtain
\[
\pi_{ik}(t) = e_k^T Z^T(t) e_i
\]
(21)

We now turn to the evaluation of the Markov diffusion distance (Equation
(18)):
\[
d_{ij}(t) = \sum_{k=1}^{n} (\pi_{ik}(t) - \pi_{jk}(t))^2 = \sum_{k=1}^{n} \left( e_k^T Z^T(t) e_i - e_k^T Z^T(t) e_j \right)^2
\]
(22)
\[
= ||Z^T(t)(e_i - e_j)||^2 = (e_i - e_j)^T Z(t) Z^T(t)(e_i - e_j)
\]
(23)

which has exactly the same form as Equation (2). We immediately deduce the
form of the Markov diffusion (\( K_{MD} \)) kernel
\[
K_{MD}(t) = Z(t)Z^T(t) \quad \text{with} \quad Z(t) = \frac{1}{t} \sum_{\tau=1}^{t} P^\tau
\]
(24)

Notice that in order to evaluate \( Z(t) \), we use a trick similar to the one used
in PageRank (Langville & Meyer, 2005): a dummy absorbing state linked to
all the states of the Markov chain is created with a very small probability of
jumping to this state. This amounts to subtracting some small quantity from
every element of \( P \), with the result that matrix \( P \) is now substochastic (if this
was not the case, \((I - P)\) would not be invertible) and that \(Z(t)\) admits the following analytical form:

\[
Z(t) = \frac{1}{t} (I - P)^{-1}(I - P^t)P
\]  

(25)

4.7 The relative-entropy diffusion matrix

Instead comparing two probability distributions with a squared distance as in Equation (18), the symmetric relative entropy is more appropriate (see, e.g., (Cover & Thomas, 2006; Kapur & Kesavan, 1992)). As it is well-known, the symmetric relative entropy (also called the Kullback-Leibler divergence) between two probability distributions \(p\) and \(q\) is defined as (Cover & Thomas, 2006; Kapur & Kesavan, 1992):

\[
D(p : q) = \sum_{k=1}^{n} \left[ p_k \log \frac{p_k}{q_k} + q_k \log \frac{q_k}{p_k} \right]
\]  

(26)

Letting \(p_k = \pi_{ik}(t)\) and \(q_k = \pi_{jk}(t)\) yields

\[
d_{ij}(t) = \sum_{k=1}^{n} \left[ \pi_{ik}(t) \log \frac{\pi_{ik}(t)}{\pi_{jk}(t)} + \pi_{jk}(t) \log \frac{\pi_{jk}(t)}{\pi_{ik}(t)} \right]
\]  

(27)

From Equation (21), \(\pi_{ik}(t) = e_i^T Z^T(t)e_i = e_i^T Z(t)e_k\), so that Equation (27) becomes

\[
d_{ij}(t) = \sum_{k=1}^{n} \left[ e_i^T Z(t)e_k \log(e_i^T Z(t)e_k) - e_i^T Z(t)e_k \log(e_j^T Z(t)e_k) \right. \\
+ e_j^T Z(t)e_k \log(e_j^T Z(t)e_k) - e_j^T Z(t)e_k \log(e_j^T Z(t)e_k) \right]
\]

\[
= \sum_{k=1}^{n} \left[ e_i^T Z(t)e_k e_i^T \log(Z^T(t))e_i - e_i^T Z(t)e_k e_j^T \log(Z^T(t))e_j \\
+ e_j^T Z(t)e_k e_i^T \log(Z^T(t))e_i - e_j^T Z(t)e_k e_j^T \log(Z^T(t))e_j \right]
\]

\[
= e_i^T Z(t) \log(Z^T(t))e_i - e_i^T Z(t) \log(Z^T(t))e_j \\
+ e_j^T Z(t) \log(Z^T(t))e_j - e_j^T Z(t) \log(Z^T(t))e_i \\
= (e_i - e_j)^T \left[ Z(t) \log(Z^T(t)) \right] (e_i - e_j) \\
= (e_i - e_j)^T \frac{Z(t) \log(Z^T(t)) + \log(Z(t))Z^T(t)}{2} (e_i - e_j) 
\]  

(28)

where we used \(\sum_{k=1}^{n} e_k e_k^T = I\) (the identity matrix) and the element-wise logarithm is taken on each element of the matrix. The last line follows from the fact that any matrix \(M\) can be expressed as the sum of its symmetric part \((M + M^T)/2\) and its asymmetric part \((M - M^T)/2\) and that the distance cancels out the asymmetric part: \((e_i - e_j)^T(M - M^T)(e_i - e_j) = 0\). Equation (28) takes the form of a Mahalanobis distance, with the important difference that \([Z(t) \log(Z^T(t)) + \log(Z(t))Z^T(t)]\) is not positive semidefinite in general.
This leads to the relative-entropy diffusion (K_{RED}) matrix:

$$K_{RED}(t) = Z(t) \log(Z^T(t)) + \log(Z(t))Z^T(t) \text{ with } Z(t) = \frac{1}{t} \sum_{\tau=1}^{t} P^\tau$$ (29)

This similarity matrix is not positive semidefinite and therefore is not a kernel.

### 4.8 The random-walk-with-restart similarity matrix

Pan et al. (Pan et al., 2006) (see also (Tong et al., 2006, 2008)) introduced a similarity matrix inspired by the well-known PageRank algorithm (Brin & Page, 1998; Page et al., 1998). This model has been applied to various interesting applications, including center-piece subgraph discovery and content-based image retrieval (Pan et al., 2006; Tong et al., 2006, 2008). The same idea was also introduced by (Pucci et al., 2006; Gori & Pucci, 2006, 2007) in the context of collaborative recommendation.

Like the diffusion kernel, the model considers a random walker jumping from some node $i$ to some neighbor node $j$ with a probability proportional to the edge weight $p_{ij} = P(s(t+1) = j | s(t) = i) = a_{ij}/a_{i\cdot}$. In addition, at each step of the random walk, the random walker has some probability $(1 - \alpha)$ of returning to node $i$ instead of continuing to neighbor nodes. In other words, the probability distribution of finding the random walker on each node at time $t$ is provided by

$$\begin{cases} x(0) = e_i \\ x(t+1) = \alpha P^T x(t) + (1 - \alpha) e_i \end{cases}$$ (30)

where $\alpha \in ]0, 1[$. Thus, random walkers have the opportunity to restart with probability $(1 - \alpha)$ from node $i$ at each time step, instead of continuing their random walk. Considering the steady-state solution $x(t+1) = x(t) = x_\infty$ and extracting the probability distribution $x$ of finding the random walker on each node when starting from node $i$ yields

$$x_\infty = (1 - \alpha)(I - \alpha P^T)^{-1}e_i$$ (31)

which corresponds, up to a scaling factor, to column $i$ of the matrix $(I - \alpha P^T)^{-1}$. Notice that since the matrix $\alpha P$ is substochastic, the inverse of $(I - \alpha P)$ exists if the Markov chain is regular. Vector $x_\infty$ can be viewed as containing a similarity between node $i$ and the other nodes of the graph.

Since $x_\infty$ is column $i$ of matrix $(I - \alpha P^T)^{-1}$, the random-walk-with-restart (K_{RWR}) matrix (Pucci et al., 2006; Gori & Pucci, 2006; Pan et al., 2006; Tong et al., 2006, 2008) can be defined as the matrix whose $i$'th row contains the similarities to node $i$ (the matrix is transposed):

$$K_{RWR} = (I - \alpha P)^{-1}$$ (32)

with the same form as the fundamental matrix $(I - P)^{-1}$ of the Markov chain.
\(K_{RWR}\) can be rewritten as
\[
K_{RWR} = (D^{-1}(D - \alpha A))^{-1} = (D - \alpha A)^{-1}D
\]  
(33)
since \(P = D^{-1}A\). Notice that \(K_{RWR}\) is not a kernel since it is not symmetric.

4.9 The regularized commute-time kernel

Remember that the Laplacian matrix, whose pseudoinverse is called the commute-time kernel in Section 4.5, is not invertible. Instead of taking the pseudoinverse of the matrix, a simple regularization framework can be applied as in Section 4.4. One such regularization leads to what we call the regularized commute-time (\(K_{RCT}\)) kernel:

\[
K_{RCT} = (D - \alpha A)^{-1}
\]  
(34)

with \(\alpha \in ]0, 1[\). Since \(K_{RCT}\) is the matrix inverse of the sum ((1 - \(\alpha\))D + \(\alpha\)L) of a positive definite matrix and a positive semidefinite matrix, it is positive definite and it is a valid kernel. This kernel is a slight extension of the modified Laplacian matrix \((\gamma D - A), \gamma \in ]0, 1[\) (Ito et al., 2005), equivalent to changing the range of \(\gamma\) to \(\gamma > 1\).

Another justification of the interest of this kernel is as follows (Mantrach et al., 2011). Consider the following random-walk model starting at node \(i\)
\[
\begin{align*}
\{x(0) &= e_i \\
x(t+1) &= P^T x(t)
\end{align*}
\]  
(35)

The column vector \(x(t)\) contains the probability distribution of finding the random walker in each state of the finite Markov chain. Thus, the random walker starts at node \(i\) and diffuses through the network. Let us define the similarity vector between node \(i\) and other nodes of the network by:
\[
\text{sim}_i = \sum_{\tau=0}^{\infty} \alpha^\tau D^{-1} x(\tau)
\]  
(36)

with \(\alpha \in ]0, 1[\). The weighting factor \(D^{-1}\) compensates for the fact that \(x(t)\) converges to the stationary distribution, which is proportional to the diagonal elements of \(D\) with undirected graphs (Ross, 1996). A similarity measure based on \(x(t)\) only (without weighting factor) would therefore favor the nodes with a high outdegree \([D]_{i*}\). Equation (36) thus accumulates, with a damping factor \(\alpha\) (later visits are less important than earlier ones), the probability of visiting each node when starting from node \(i\). From another point of view, it also corresponds to an “evaporating” random walk where the random walker has a \((1 - \alpha)\) probability of disappearing at each step. Equation (36) cumulates the presence rate in each node during this evaporating random walk, when starting from node \(i\).
Table 1
The nine kernels with the value range tested for their parameters.

We easily find

\[ \text{sim}_i = \sum_{\tau=0}^{\infty} \alpha^\tau D^{-1} x(\tau) = D^{-1} \sum_{\tau=0}^{\infty} \alpha^\tau (P^T)^\tau e_i \]
\[ = D^{-1} (I - \alpha P^T)^{-1} e_i = D^{-1} \left( (D - \alpha A)D^{-1} \right)^{-1} e_i \]
\[ = (D - \alpha A)^{-1} e_i = \text{col}_i(K_{RCT}) \]

where we used \( P = D^{-1/2} \) and the fact that \( D \) is diagonal and \( A \) is symmetric (undirected graph).

Thus, the \((i, j)\) element of \( K_{RCT} \) can be interpreted as the discounted cumulative probability of visiting node \( j \) when starting from node \( i \). The regularized commute-time kernel (Equation (34)) differs from the random-walk-with-restart similarity matrix (Equation (33)) only in the fact that the latter post-multiplies the former by \( D \). The regularized commute-time kernel matrix was already presented in (Zhou et al., 2003) in the context of semisupervised classification, through a regularization framework along the same lines as in Equation (12).

4.10 Parameter tuning and efficiency

Table 1 lists the kernel-based algorithms. The “Equation” column refers to the number in this paper of the equation used in the experiments to compute each kernel (those equations are bowed in the text). Most algorithms need a value for some parameters (given in the “Par.” column). Those values (see the “Tested values” column) were tuned in the experiments following a nested cross-validation procedure detailed in Section 5.3 for recommendation and in Section 6.2 for classification.

Efficiently inverting matrices can have a significant influence on performance for large graphs. Also, inverse matrices are often dense and may not fit in main memory. Cholesky factorizations address those issues (see, e.g., (Fouss et al., 2007)). Incomplete (lower-rank) Cholesky matrix factorizations (see, e.g., (Fine & Scheinberg, 2001)) can be efficiently computed and remain sparse if the original matrix is sparse. From the factorization, every column of the matrix inverse can be obtained by simple backsubstitution. Therefore, a recommendation to a specific user or the classification of an unlabeled node of interest
by kernel alignment can be obtained quite efficiently from the factorization. Notice also that columns of the kernel matrix can be obtained by solving a linear system of equations and that this does not require the computation of the inverse. However, kernel-based methods are not suited for large-scale graphs (millions of nodes) since they require a matrix inverse of factorization.

5 Testing the kernels on collaborative recommendation

This section reports on experimental work aiming at (i) comparing the accuracy of the various kernels and (ii) comparing their performance with that of the latent-class model (Cheung et al., 2004; Hofmann, 2004), a well-established and effective recommendation method. Recommender systems provide their users with ranked lists of items that they will appreciate, based on past preferences, history of choices, and demographic information. For example, imagine a simple movie database with three sets of elements people, movie, and movie_category, and two relationships has_watched, between people and movie, and belongs_to, between movie and movie_category. Computing similarities between people and movies allows to recommend lists of movies for people to watch. The idea originates (see the survey in (Adomavicius & Tuzhilin, 2005)) mainly from research on machine learning, information retrieval (Salton, 1989), cognitive science (Rich, 1979), forecasting theories (Armstrong, 2001), marketing (Lilien, Smith, & Moorthy, 1992), and management (Murthi & Sarkar, 2003). Recommender systems emerged as a research area in the mid-1990s, with the first papers on collaborative filtering (Hill, Stead, Rosenstein, & Furnas, 1995; Resnick, Neophytos, Mitesh, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995).

The nine algorithms presented in Section 4 were tested on a collaborative-recommendation task with two concrete datasets. A weighted undirected graph is associated with a database in the following way: database elements correspond to nodes of the graph and database links correspond to edges. In the movie database, people and movies correspond to nodes of the (bipartite) graph, and has_watched links appear as edges connecting the corresponding nodes. In the corresponding user-item matrix, the weight of an edge is set to 1 if there is a link between the corresponding items and to 0 otherwise. Recommending items to a user amounts to predicting missing user-item links in the graph. Matrices for typical recommendation applications are sparse. Therefore the assumption that the corresponding graph is connected may not apply.

5.1 Datasets

With the MovieLens (ML) web-based recommender system\(^2\), visitors of the web site are invited to rate movies that they watched and to ask for recommendations about movies that they should watch, given those that they and other users have already watched. We selected a sample of the ML database as suggested in (Sarwar, Karypis, Konstan, & Riedl, 2002). Enough users (i.e., 943 of them) were randomly selected to obtain 100,000 ratings, retaining only

\(^2\) http://www movielens.umn.edu
users who had rated at least 20 movies on a total of 1,682 movies. The resulting user-movie matrix contains about 6.3% of “1” values and 93.7% of “0” values.

The Book-Crossing (BC) used in this paper (collected in 2004 from the Book-Crossing community) contains 278,858 users providing 1,149,780 ratings about 271,379 books. We selected from it a sample of 2,222 books, 1,028 users, and 109,374 ratings, retaining only users who had rated at least 40 books and books that had been rated by at least 20 users. The user-book matrix contains about 4.8% of “1” values and 95.2% of “0” values, which makes it sparser than the matrix created from the MovieLens database.

Our experiments discarded the numerical ratings provided in the datasets (that is, we retained only the fact that a person did or did not rate an item). Indeed, few numerical ratings are provided for the BC dataset while most ratings in the ML dataset are 4's or 5's on a 1-to-5 preference scale, which reduces their significance for discrimination. The graphs computed from the preprocessed datasets are connected. Isolated users or items appearing during the cross-validation splits (see Section 5.3) were automatically removed in order to work with connected graphs.

5.2 Direct and indirect recommendation

We used three methods to test the kernels. We call them, respectively, the “direct method”, the “user-based indirect method”, and the “item-based indirect method”. Both indirect methods are based on a nearest-neighbor technique that requires a measure of closeness or similarity. Important considerations involved in choosing a similarity measure (Johnson & Wichern, 2002) include the nature of variables (discrete, continuous, or binary), the scale of measurements (nominal, ordinal, interval, or ratio), and specific knowledge about the subject matter. The nearest-neighbor scoring algorithm is one of the simplest and oldest methods for general classification tasks (Duda, Hart, & Stork, 2001). It classifies an unknown pattern as belonging to the class of the nearest example in the dataset as measured by a similarity metric. K-nearest-neighbor techniques generalize the idea to taking into account k nearest examples to decide on class membership.

The direct method computes similarities sim(p0, i) with each scoring algorithm between a given user p0 requesting recommendations and items i. As an answer, a ranked list of items is suggested to user p0 from among those that p0 did not rate.

The user-based indirect method suggests items by applying a two-step process:

1) identify users with rating patterns most similar to those of user p0 requesting recommendations;

The user-based indirect method suggests items by applying a two-step process:
1) identify users with rating patterns most similar to those of user p0 requesting recommendations;

http://www.informatik.uni-freiburg.de/~ziegler/BX/
http://www.bookcrossing.com
use the ratings from those users to compute recommendations.

More precisely, the user-based indirect method uses the scoring algorithms like the direct method to compute similarities \( \text{sim}(p_0, p) \) between user \( p_0 \) and users \( p \). Then, it computes, from the \( k \) nearest-neighbor users of \( p_0 \), the preference or predicted value for each item. The preference for item \( i_0 \) and user \( p_0 \) is the sum, weighted by \( \text{sim}(p_0, p) \), of the values of the link weight (0 or 1) of item \( i_0 \) for the \( k \) users \( p \) closest to \( p_0 \) (\( k \) nearest neighbors):

\[
pred(p_0, i_0) = \frac{\sum_{p=1}^{k} \text{sim}(p_0, p) \ w_{pi0}}{\sum_{p=1}^{k} \text{sim}(p_0, p)} \tag{40}
\]

where \( w_{pi0} \) is “1” if user \( p \) rated item \( i_0 \) and “0” otherwise. The higher the preference \( \text{pred}(p_0, i_0) \), the stronger the recommendation that \( p_0 \) will like \( i_0 \).

The **item-based indirect method** works as follows:

1. first, for each item, determine a set of its nearest neighbor items;
2. use these neighborhoods and the ratings of the requesting user \( p_0 \) to compute recommendations.

This method corresponds to the methodology proposed by the SUGGEST approach (Karypis, 2001). It first applies scoring algorithms to compute similarities \( \text{sim}(i_0, i) \) between a given item \( i_0 \) and other items \( i \). Then, for user \( p_0 \) and item \( i_0 \), it computes, from the \( k \) nearest items of \( i_0 \) (\( k \) nearest neighbors), the preference or predicted value of \( i_0 \) for \( p_0 \) as a sum, weighted by \( \text{sim}(i_0, i) \), of the values of the link weight (0 or 1) of \( p_0 \) and items \( i \):

\[
pred(p_0, i_0) = \frac{\sum_{i=1}^{k} \text{sim}(i_0, i) \ w_{pi0}}{\sum_{i=1}^{k} \text{sim}(i_0, i)} \tag{41}
\]

where \( w_{pi0} \) is defined as above. Items are suggested to user \( p_0 \) in decreasing order of predicted values, among those not rated by \( p_0 \).

**Choosing a value for \( k \).** When applying an indirect method to compute a ranked list of items to be suggested to a user, a value for the number \( k \) of neighbors must be chosen. For each scoring algorithm and both indirect methods, we systematically varied \( k \) in the range (10, 20, ..., 100). We decided to limit the maximum number of neighbors to 100, a fairly large number. Since the neighbors are sorted by decreasing similarity, as explained earlier in this section, the weight given to the \( i^{th} \) neighbor (a user or an item, depending on the method) corresponds to a similarity value (as computed by the scoring algorithm) that is higher than the weight given to the \( (i + 1)^{th} \) neighbor. In other words, the farther the neighbor, the smaller its contribution to the final predicted value (as computed by Equations (40) and (41)). Parameter \( k \) is tuned by using an internal cross-validation (see Section 5.3 below).
The performance of the various scoring algorithms is evaluated by applying a standard nested (or double) cross-validation based on links. This involves tuning parameters, namely, the parameter of the kernels (see Section 4), the number of neighbors (see Section 5.2), and the number of latent classes (see Section 5.4). Evaluation consists in determining the measure providing the best results (see Section 2.2) among (1) the original kernel, (2) the centered kernel matrix, (3) the cosine kernel matrix, (4) the centered cosine kernel matrix, and (5) the distance measure. This evaluation is performed with an internal nine-fold cross-validation and performance is averaged on an external ten-fold cross-validation.

More precisely, each dataset (containing links between users and items) was divided into ten subsets of links. Each subset (i.e., ten percent of the links) was in turn used for performance estimation in the external cross-validation (it is referred to as the “external test set”). The nine remaining subsets, merged to form the “external training set”, were used for the internal cross-validation to tune the parameters. On these nine subsets, an internal cross-validation was performed mirroring the external cross-validation, i.e., one of these nine subsets was used in turn as an “internal test set” while the remaining data subsets were merged, forming an “internal training set”. We chose as values for the parameters those providing the best average performance on the internal nine-fold cross-validation and we used in turn those values for performance evaluation on the external ten-fold cross-validation.

For indirect methods, we used two ways to work with the original kernel matrix: (1) apply the transformation to the whole kernel matrix (for example, centering), extract the user-user matrix or the item-item matrix, and proceed from there, and (2) extract the user-user matrix or the item-item matrix from the original kernel matrix, apply the transformation (for example, centering) on this submatrix, and proceed from there. Both ways were tested in the nested cross-validation procedure.

Thus, each scoring algorithm provides, for each person, a set of proximity measures indicating preferences about items. In particular, for kernels, the kernel matrix is computed from the adjacency matrix of the bipartite graph containing person and item nodes, and containing a \((i, j)\) link if user \(i\) rated item \(j\). From that information and for each person, we extracted a ranked list of all the items that the person has not yet rated, according to the training set. In that list, the items closest to the person, in terms of the proximity measure, are considered the most relevant.

**Accuracy.** To evaluate the accuracy of the scoring algorithms, we compared their performance by applying a recall measure (Herlocker, Konstan, Terveen, & Riedl, 2004). For each user, the test set (external or internal) contains a set of items that the user actually rated and that were removed from the training set. Those items are part of the ranked list supplied by each scoring algorithm which contains all the items that the person has not yet rated, according to the training set. Ideally, these items should be ranked first since they were rated by the user. The recall accuracy measure (called “recall score” in the
sequel) was used to compare the ranking provided by each algorithm (where all the items are ranked) and the “ideal ranking” (where each item from the test set is ranked before an item not belonging to the test set).

More precisely, the recall score is the average (on all users) of the percentage of items from the test set that appear among the top k of the ranked list, for some given k. A recall score of 100% indicates that the scoring algorithm always positions the items in the test set among the top k of the ranked list (assuming that the number of items in the test set is smaller than k). This measure should be as high as possible for good performance. We report the recall for the top 10 (Recall 10) and the top 20 (Recall 20) items (for a total of 1,682 items in the ML dataset and 2,222 books in the BC dataset).

We did not consider the case where new elements (i.e., new users or new items) are added into the dataset: only the case where “new” links (forming the test set) are added is considered. Iterative procedures could be developed to this aim for the various scoring algorithms.

5.4 Other (non kernel-based) scoring algorithms

The kernel-based algorithms were compared to four standard techniques: the baseline (Basic) algorithm, the standard binary algorithm (Bin), the cosine algorithm (Cos), and the latent-class algorithm (LCl). The latter three algorithms were chosen for comparison since they provide the best results among a number of standard methods (Fouss et al., 2007): Katz’ method, shortest-path algorithm, maximum flow algorithm, and correspondence analysis. More detail about these methods as well as their results obtained on the MovieLens dataset are presented in (Fouss et al., 2007). The last three algorithms depend on a parameter that is optimized in the inner loop of the nested cross-validation.

The baseline algorithm (Basic) serves as a reference for comparison.

1. In its direct form (see Section 5.2), Basic simply recommends items according to the number of times that they have been rated. Thus the ranking is the same for all users. This is similar to basing decisions only on a-priori probabilities in supervised classification.

2. In its user-based indirect form, the user-user similarity matrix to recommend items is simply the usual “bibliographic-coupling matrix” $W^T W$ of bibliometry (Ito et al., 2005; Kessler, 1963) where $W$ is the user-item adjacency matrix. Two users are similar if the corresponding element of $W^T W$ is high, i.e., if there are many items that have been rated by both users. From these similarity measures (sim($i_k$, $i_l$) = element $k,l$ of $W^T W$), a ranked list of items is then computed with Equation (40).

3. Symmetrically, in its item-based indirect form, the item-item similarity matrix to recommend items is the “co-citation matrix” $W^T W$. Two items are similar if the corresponding element of $W^T W$ is high, i.e., if there are many users who have rated both items. From these similarity measures (sim($i_k$, $i_l$) = element $k,l$ of $W^T W$), a ranked list of items is then computed with Equation (41).
**Binary algorithm (Bin).** Pairs of items are compared on the basis of the presence or absence of certain features (e.g., rating a particular movie or a particular book). The presence or absence of a feature can be described with a binary variable, with value 1 if the feature is present (i.e., person \(i\) rated item \(j\)) and value 0 if the feature is absent (i.e., person \(i\) did not rate item \(j\)). We now detail the procedure followed by applying Bin in the user-based indirect method (see Section 5.2); applying Bin in the item-based indirect method is similar and does not require further explanations.

Each user \(i\) is characterized by a binary vector \(v_i\) (whose dimension is the total number of items) encoding the items rated by that user. The \(k\) nearest neighbors of person \(i\) are extracted by taking the \(k\) nearest \(v_j\) according to some binary similarity measure \(\text{sim}(i, j) = s(v_i, v_j)\). In (Fouss et al., 2007), we performed systematic comparisons between eight such measures \(s\) (listed in (Johnson & Wichern, 2002), p. 674) for different values of \(k\) (= 10, 20, ..., 100). The best recall scores were obtained with the measure “ratio of 1-1 matches to mismatches with 0-0 matches excluded”. Notice that Bin can only be used as an indirect method with bipartite graphs. The number of nearest neighbors is optimized in the inner loop of the nested cross-validation.

**Cosine algorithm (Cos).** The cosine coefficient between variables \(i\) and \(j\) measures the strength and the direction of a linear relationship between two variables. It is defined as \(\text{sim}(i, j) = \cos(i, j) = (v_i^T v_j)/(|v_i| |v_j|)\) (Dunham, 2003; Sarwar, Karypis, Konstan, & Riedl, 2001). If variables are items (i.e., considering the item-based indirect method based on finding neighbors to items) represented as vectors in the user space (item \(i\) is characterized by vector \(v_i\)), two items are considered more similar if the angle between their corresponding vectors in the user space is smaller. Like Bin, Cos is an indirect method that can only be used with bipartite graphs. The number of nearest neighbors is optimized in the inner loop of the nested cross-validation.

**Latent-class algorithm (LCl).** Latent-variable models provide an important tool for the analysis of multivariate data. Latent variables appear in many fields to which statistical methods are applicable, especially in social sciences (Bartholomew, 1999). Many patterns (represented by latent variables) that are of interest to social scientists cannot be observed directly. Examples are preferences, attitudes, behavioral intentions, and personality traits. Such parameters can only be estimated indirectly by means of observable indicators, such as questionnaires designed to extract information on, for example, a specific attitude or preferences of a person.

A latent-variable model, called “latent-class model”, was developed in the context of collaborative filtering. It can be considered a sound and state-of-the-art recommendation technique (Cheung et al., 2004; Hofmann, 2004). Basically, it is a clustering model assuming that the preferences of a user are established through a latent variable. In this model, a latent-class variable \(z \in Z = \{z_1, ..., z_q\}\) (\(q\) being the number of latent classes or patterns) is associated with each observation \((x, y)\) where \(x\) represents a user and \(y\) an item. The key assumption of the latent-class model is that \(x\) and \(y\) are independent, conditional on \(z\). The standard procedure for maximum likelihood estimation in latent-class models is the Expectation-Maximization (EM) algorithm (introduced in (Dempster, Laird, & Rubin, 1977)). The number of latent classes
is optimized in the inner loop of the nested cross-validation.

5.5 Results and discussion

Thus, for each person and each cross-validation split, we first selected the items that were not rated (according to the training set). Then, we ranked them with each scoring algorithm. Finally, we compared those rankings with the test set by using the recall score described in Section 5.3.

Results are summarized in Table 2 (for the MovieLens dataset) and Table 3 (for the Book-Crossing dataset), which show the recall, considering either the top 10 of the ranked list (Recall 10) or the top 20 of the ranked list (Recall 20) for the best configuration (in terms of the parameter of the scoring algorithm, the number of neighbors, and the applied derived measure) of each of the thirteen scoring algorithms (nine kernels on a graph, and Bin, Cos, Basic and LCI). The standard deviation of the results (STD) across the ten external cross-validation sets is also reported. The smaller the STD is, the more stable is the performance of the algorithm across different data samples. We used a paired t-test to determine whether there is a significant difference (with a p-value smaller than 0.05) between the results of the algorithms. The best results overall, for each measure of performance, are displayed in bold, based on the t-test.

5.5.1 MovieLens dataset.

Table 2 shows that, with the direct method to rank movies for each user, the best Recall 10 is obtained by K\textsubscript{MD} (20.73) while the best Recall 20 is obtained by LCI (31.55). For the user-based indirect method, the best results are obtained by K\textsubscript{CT} and K\textsubscript{MD} for both recalls (21.41, 32.09 and 21.40, 31.98, respectively – differences are not significant according to the paired t-tests). For the item-based indirect method, the best results are obtained by K\textsubscript{RED} and K\textsubscript{MD} for Recall 10 (19.85 and 19.72 respectively) and by Bin for Recall 20 (31.16), followed by K\textsubscript{RED} (30.63), and K\textsubscript{MD} (30.58).

Overall, for all three methods (see results in bold), the best scores are obtained by K\textsubscript{CT} (21.41 for Recall 10 and 32.09 for Recall 20) and K\textsubscript{MD} (21.40 for Recall 10 and 31.98 for Recall 20). The best recall scores for a standard scoring algorithm are obtained by Cos in the user-based indirect method (20.70 for Recall 10 and 31.18 for Recall 20), closely followed by Bin.

5.5.2 Book-Crossing dataset.

Table 3 shows that, with the direct method, the best results are obtained by K\textsubscript{RCT}, K\textsubscript{RL}, and K\textsubscript{CT} for both recalls (7.35, 7.36, and 7.26, respectively, for Recall 10, and 11.74, 11.58, and 11.73, respectively, for Recall 20). For the user-based indirect method, the best results are obtained by four kernels (i.e., K\textsubscript{RL}, K\textsubscript{RCT}, K\textsubscript{CT}, and K\textsubscript{MD}) for both recalls (7.09, 7.12, 7.07, and 7.09, respectively, for Recall 10; 11.23, 11.39, 11.39, and 11.31, respectively, for Recall 20). For the item-based indirect method, the best recall scores are obtained by K\textsubscript{CT}
it. Remember that Bin and Cos can only be used in indirect methods, and LCl datasets, the best score for each algorithm and the method applied to obtain it. Overall (see results in bold), the best scores are obtained by the KCT kernel and the Bin standard scoring algorithm, with the item-based indirect method.

| Recall 10 | 18.52 | 18.06 | 15.05 | 18.94 | 19.75 | 17.24 | 20.73 | 20.39 | 16.57 | / | / | 10.98 | 20.29 |
| Recall 20 | 27.84 | 26.95 | 23.05 | 28.38 | 30.29 | 26.98 | 30.96 | 30.70 | 25.99 | / | / | 17.41 | 31.55 |

Table 2
Average performance, on the MovieLens dataset, for the thirteen scoring algorithms and the three methods (direct, user-based indirect, and item-based indirect) for using them.

| Recall 20 | 29.93 | 31.09 | 29.83 | 31.70 | 31.28 | 32.09 | 31.98 | 30.07 | 30.68 | 31.07 | 31.18 | 28.34 | / |

Table 3
Average performance, on the Book-Crossing dataset, for the thirteen scoring algorithms and the three methods (direct, user-based indirect, and item-based indirect) for using them.

(8.84 for Recall 10 and 13.58 for Recall 20) together with a standard scoring algorithm, Bin (8.79 for Recall 10 and 13.50 for Recall 20).

Overall (see results in bold), the best scores are obtained by the KCT kernel and the Bin standard scoring algorithm, with the item-based indirect method.

5.5.3 General discussion.
Table 4 shows, for the four settings (i.e., Recall 10 and Recall 20 for both datasets), the best score for each algorithm and the method applied to obtain it. Remember that Bin and Cos can only be used in indirect methods, and LCl only directly.

For the ML dataset, with the exception of KRED (for both recalls) and Bin (for Recall 20), the best results (in 21 of 24 cases – ignoring LCl that can be used
the last four algorithms are below average: K
above-average (i.e., positive) CR scores: K
Table 5 suggests that the best kernel-based algorithm (in terms of average CR
“CR ranking” row ranks the algorithms according to these average scores.
The “Average CR score” row is the average of the first twelve results. The
Table 5 attempts a fair comparison between the nine kernel-based algorithms
This interesting observation suggests a possible link between the sparsity of
We observe that K_CT produces the best scores in all four settings, virtually
able to outperform the standard methods, which is a bit disappointing. While the results show that the kernels
methods accurately capture some structural information, they do not provide
a significant added value over standard methods.
Table 5 attempts a fair comparison between the nine kernel-based algorithms
by normalizing the numbers (the “initial scores”) in Table 4. The numbers
in each row i of the twelve first rows of Table 5 are computed as follows (the
“baseline score” is the score obtained by the Basic algorithm):

(1) new score = (initial score - baseline)/baseline.
(2) final score = (new score – average(i))/std(i), where average(i) is the mean
score on row i and std(i) is the standard deviation on row i).

The “Average CR score” row is the average of the first twelve results. The
“CR ranking” row ranks the algorithms according to these average scores.
Table 5 suggests that the best kernel-based algorithm (in terms of average CR
score) is K_MD (0.97), followed by K_RCT (0.77), and by K_CT (0.70). A paired
t-test shows that there is no significant difference (with a p-value smaller than
0.05) between K_MD and K_CT ranges of scores. Two other algorithms exhibit
above-average (i.e., positive) CR scores: K_RL (0.54) and K_RE (0.28), while
the last four algorithms are below average: K_ED (–0.96), K_LED (–0.62), K_VND

<table>
<thead>
<tr>
<th>MovieLens dataset</th>
<th>K_ED</th>
<th>K_LED</th>
<th>K_VND</th>
<th>K_RL</th>
<th>K_RCT</th>
<th>K_CT</th>
<th>K_MD</th>
<th>K_RE</th>
<th>K_RW</th>
<th>Bin</th>
<th>Cos</th>
<th>Basic</th>
<th>LCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>dir</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
</tr>
<tr>
<td>Recall 20</td>
<td>29.93</td>
<td>31.09</td>
<td>29.81</td>
<td>31.70</td>
<td>31.28</td>
<td>32.09</td>
<td>31.98</td>
<td>30.70</td>
<td>30.68</td>
<td>31.16</td>
<td>31.18</td>
<td>26.34</td>
<td>31.55</td>
</tr>
<tr>
<td>method</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>dir</td>
<td>user</td>
<td>item</td>
<td>item</td>
<td>user</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Book-Crossing dataset</th>
<th>K_ED</th>
<th>K_LED</th>
<th>K_VND</th>
<th>K_RL</th>
<th>K_RCT</th>
<th>K_CT</th>
<th>K_MD</th>
<th>K_RE</th>
<th>K_RW</th>
<th>Bin</th>
<th>Cos</th>
<th>Basic</th>
<th>LCI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall 10</td>
<td>6.45</td>
<td>6.33</td>
<td>6.42</td>
<td>8.40</td>
<td>8.72</td>
<td>8.84</td>
<td>8.70</td>
<td>8.62</td>
<td>8.58</td>
<td>8.79</td>
<td>8.03</td>
<td>5.32</td>
<td>5.83</td>
</tr>
<tr>
<td>method</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>dir</td>
</tr>
<tr>
<td>method</td>
<td>user</td>
<td>user</td>
<td>user</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>item</td>
<td>dir</td>
</tr>
</tbody>
</table>

Table 4
Summary of the best accuracy results, for each scoring algorithm and both recall
scores, with the method providing the best result.

only by the direct method) are obtained with the user-based indirect method.
For the BC dataset, with the exception of K_ED, K_LED, and K_VND (all three
for both recalls), the best results (in 18 of 24 cases – ignoring again LCI) are
obtained with the item-based indirect method.

We observe that K_CT produces the best scores in all four settings, virtually comparable with K_MD on both settings with the ML dataset and Bin on both
settings with the BC dataset. The results also indicate that the best kernels
on graphs are competitive with the standard algorithms. For example, K_MD, K_RCT obtain slightly better results than the Bin method that performs best among the standard algorithms. However they do not outperform the standard
methods, which is a bit disappointing. While the results show that the kernels
methods accurately capture some structural information, they do not provide
a significant added value over standard methods.
(-1.30), and $K_{RWR}$ (-0.38). Notice that $K_{MD}$, $K_{RCT}$, and $K_{RL}$ are the only kernels with all their scores positive, providing argument for the stability of these kernels. $K_{MD}$ and $K_{RCT}$ can therefore be considered as the best kernels on these collaborative recommendation tasks.

### 6 Testing the kernels on semisupervised classification

This section reports on another experimental comparison of the accuracy of the various kernels on a semi-supervised classification task. Their performance was also compared to that of a state-of-the-art semisupervised random-walk-based classification method, namely the $D$-walks model (Callut, Francoisse, Saerens, & Dupont, 2008). In a practical classification setting, labeled data is typically hard to obtain, since it often depends on expensive human labor. Unlabeled data is more easily available. For example, large databases of text documents are readily available but explicitly classified text databases are much scarcer. Semisupervised classification algorithms take advantage of whatever labeled data is available to improve the classification accuracy. For a comprehensive survey of semisupervised classification, see (Abney, 2008; Chapelle et al., 2006; Zhu, 2008; Zhu & Goldberg, 2009).

The main objective is to infer labels for the unlabeled nodes from the labeled ones (Macskassy & Provost, 2007), that is, to classify unlabeled nodes. In addition to available node labels, the graph topology provides useful information. Several values of the labeling rate (i.e., the proportion of nodes with known labels) were considered (ranging from 10% to 90%). In order to assess the ability of the algorithms to determine correct labels, some labels were removed from the original dataset and the removed data was used as a test set (the remaining data constituting the so-called training set). For each labeling rate considered, 50 random deletions of labels were performed thus creating 50 test sets of unlabeled nodes on which performance was averaged.
Datasets, seven in total, comprise partially graphs (NewsGroup, Cora, WebKB, IMDb) described below. They are real-world representations of networked data. They come from several domains that have been the subject of prior study in machine learning (Macskassy & Provost, 2007). For each dataset, local consistency of node labeling is assumed, i.e., it is assumed that nodes within the same neighborhood tend to share the same label. This could be assessed by using autocorrelation indexes, such as Moran’s $I$ or Geary’s $C$, as proposed in the spatial statistics literature (Cressie, 1993; Haining, 2003).

The collaborative Internet Movie Database (IMDb) has had several applications, like making recommendations for movies to watch or classifying movies into categories. Here, the classification problem deals with predicting movie popularity (whether a movie is a likely blockbuster, award winner, b-movie, or stinker). It exploits a graph of movie nodes linked when they share a production company. Edge weights denote the number of production companies that two movies have in common. The dataset is summarized in Table 6.

The Cora dataset is a graph of 3,583 nodes collected from research papers on machine learning labeled with one of seven different topics. Two papers are linked if they share an author or if one paper cites the other one. The composition of the dataset is summarized in Table 7.

The WebKB data consists of four datasets of web pages gathered from four university departments, with each page manually labeled as belonging to one of six authorship categories: course, department, faculty, project, staff, and student. Pages are linked by co-citation ($x$ and $y$ are linked if $x$ references $z$ and $y$ references $z$). The composition of the dataset is summarized in Table 8.

The NewsGroup dataset\(^5\) comprises about 20,000 unstructured documents, taken from 20 Usenet newsgroups. News articles are labeled as belonging to one of five categories out the 20 original ones. The graph of documents was built by sampling about 200 documents at random from five different topics (see Table 9). The links between documents were extracted from the five samples by eliminating stopwords, by stemming using a TF-IDF (term frequency-inverse document frequency) technique, by computing mutual information, and finally by computing the document-document matrix, as described in (Yen et al., 2007). The composition of the dataset is summarized in Table 9.


### Table 6

<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-revenue</td>
<td>572</td>
</tr>
<tr>
<td>Low-revenue</td>
<td>597</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>1,169</td>
</tr>
<tr>
<td><strong>Majority class proportion</strong></td>
<td>51.1%</td>
</tr>
<tr>
<td><strong>Number of Edges</strong></td>
<td>40,564</td>
</tr>
</tbody>
</table>

The **IMDb** dataset.

6.1 Datasets
<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-based</td>
<td>402</td>
</tr>
<tr>
<td>Genetic algorithms</td>
<td>551</td>
</tr>
<tr>
<td>Neural networks</td>
<td>1,064</td>
</tr>
<tr>
<td>Probabilistic methods</td>
<td>529</td>
</tr>
<tr>
<td>Reinforcement learning</td>
<td>335</td>
</tr>
<tr>
<td>Rule learning</td>
<td>230</td>
</tr>
<tr>
<td>Theory</td>
<td>472</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>3,583</strong></td>
</tr>
</tbody>
</table>

**Majority class proportion** 29.7%

**Number of Edges** 22,516

Table 7
The CORA dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Cornell</th>
<th>Texas</th>
<th>Washington</th>
<th>Wisconsin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course</td>
<td>54</td>
<td>51</td>
<td>170</td>
<td>83</td>
</tr>
<tr>
<td>Department</td>
<td>25</td>
<td>36</td>
<td>20</td>
<td>37</td>
</tr>
<tr>
<td>Faculty</td>
<td>62</td>
<td>50</td>
<td>44</td>
<td>37</td>
</tr>
<tr>
<td>Project</td>
<td>54</td>
<td>28</td>
<td>39</td>
<td>25</td>
</tr>
<tr>
<td>Staff</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Student</td>
<td>145</td>
<td>163</td>
<td>151</td>
<td>155</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>346</strong></td>
<td><strong>334</strong></td>
<td><strong>434</strong></td>
<td><strong>348</strong></td>
</tr>
</tbody>
</table>

**Majority class proportion** 41.9% 48.8% 39.2% 44.5%

**Number of Edges** 26,832 32,988 30,462 33,250

Table 8
Composition of the WebKB dataset.

<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer/graphics</td>
<td>200</td>
</tr>
<tr>
<td>Computer/pchardware</td>
<td>200</td>
</tr>
<tr>
<td>Motor/auto</td>
<td>200</td>
</tr>
<tr>
<td>Religion/atheism</td>
<td>200</td>
</tr>
<tr>
<td>Politics/mideast</td>
<td>200</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1,000</strong></td>
</tr>
</tbody>
</table>

**Majority class proportion** 20.0%

**Number of Edges** 328,904

Table 9
Composition of the Newsgroup dataset.
The kernels studied in this paper are compared on the seven datasets just introduced with the following methodology. The classification of unlabeled nodes is performed with a simple kernel-alignment strategy based on a sum of similarities. More precisely, each class $c$ and its nodes are described by an $n$-dimensional indicator vector $y^c$ containing a “1” entry when the corresponding node belongs to class $c$ and a “0” entry otherwise, that is, when the node is unlabeled or belongs to another class. The similarity of node $i$ with the set of nodes belonging to class $c$ is the value of the $i$-entry of the column vector $K_y^c$ where $K$ is the kernel matrix. This similarity is computed for each class $c$ in turn. Then, each node is assigned to the class with largest similarity. The column vector containing the estimated class index of all nodes is thus $\hat{y}^* = \arg\max_c (K_y^c)$. For the kernels that are inverses of matrices, if only one node is to be labeled, a number of systems of linear equations can be solved instead of inverting the matrix (Mantrach et al., 2011). As discussed in Section 4.4, this simple kernel alignment rule is based, for some particular kernels, on the optimization of a loss function taking available class labels and smoothness into account (Belkin et al., 2004; Belkin, Niyogi, & Sindhwani, 2006; Wang et al., 2009; Yajima & Kuo, 2006; Zhou et al., 2003, 2005; Zhu et al., 2003). For a theoretical analysis of this regularization framework, see (Johnson & Zhang, 2007, 2008).

This classification methodology is arguably somewhat primitive, but the simple decision rule permits to check which kernels accurately capture similarity and class membership. The task thus consists in classifying unlabeled nodes – classifiers assign the most appropriate class to each unlabeled node. The classification accuracy was computed for several values of the labeling rate (10, 20,..., 90 percents), i.e., the proportion of nodes with known label. The labels of remaining nodes were removed and the corresponding sets of nodes were used as test sets. For each labeling rate, 50 random deletions of node labels (50 runs) were performed and classification rate was averaged on those 50 runs.

For each run, a ten-fold internal cross-validation was performed in order to tune the parameters of each classifier (typically, the parameters of the kernels, as detailed in Table 1). The performance on each run was assessed on the remaining unlabeled nodes with the tuned parameter.

### 6.3 Other (non kernel-based) scoring algorithms

The kernel-based alignment procedure was compared to two algorithms, NetKit (Macskassy & Provost, 2007), as a baseline, and the $D$-walks approach (Callut et al., 2008).

**NetKit** is based on a relational learning approach (Macskassy & Provost, 2007). This general framework builds a model based on three components: a local classifier to generate class priors, a relational classifier, which relies on the relations in the network to guess the class membership, and a so-called
collective inferencing component which further refines the class predictions. The main advantage of this framework is that each of the three components can be instantiated with various existing methods making it easily adaptable to many situations. For our experiments, testing all the module configurations would have been prohibitive. For the baseline, we chose the parameters that generally provide the best results and are quite robust according to (Macskassy & Provost, 2007). More precisely, the local classifier inducer uses the class prior, the relational classifier inducer uses the "weighted vote relational neighbor classifier", and a relaxation labeling is used for the collective inferencing. The first two instantiations do not take any parameter as they imply only class priors and direct nearest neighbors (no descriptor on the nodes). As for the relaxation labeling (Chakrabarti, Dom, & Indyk, 1998), parameters were set to default values (described in Section 4.2.2 of (Macskassy & Provost, 2007)), that were shown to be quite robust (Macskassy & Provost, 2007).

\textbf{D-walks} rely on bounded random walks performed on the input graph seen as a Markov chain (Callut et al., 2008). A betweenness measure, based on passage times during random walks of bounded length, is derived for each class (label category). Unlabeled nodes are assigned to the category for which the betweenness is highest. This approach has a linear time complexity with respect to the number of edges, the maximum walk length considered, and the number of classes; such a low complexity allows to deal with very large graphs while remaining competitive with other techniques. It has a unique hyper-parameter, the walk length, that can be tuned during cross-validation.

![Classification rate on cora](image)

Figure 1. Average classification rates for the \textbf{CORA} dataset as a function of labeling rate.

6.4 Results and discussion

Classification results for each kernel, and for NetKit and \textbf{D-walks} on all datasets are shown in Figures 1 to 7. Some methods that perform poorly do not appear

35
Figure 2. Average classification rates for the **IMDb** dataset as a function of labeling rate.

Figure 3. Average classification rates for the **WebKB Cornell** dataset.

in the charts. Thus results for methods $K_{ED}$ and $K_{RED}$ in Figure 1 and those for methods $K_{ED}$, $K_{VND}$, and NetKit in Figure 2 are not shown. The same holds for method $K_{RED}$ in both Figure 3 and Figure 4, for method $K_{RED}$ in Figure 5, for methods $K_{RED}$, $K_{ED}$, and $K_{LED}$ in Figure 6, and for method $K_{ED}$ in Figure 7. Note that the curves for $K_{RED}$ and $K_{ED}$, as well as for $K_{RCT}$ and $D$-walks, overlay in Figure 1, like those of $K_{LED}$ and $K_{ED}$ in Figure 3.

To facilitate comparisons, a summary table containing the relative “area under curves” (AUC score)\(^6\) of Figures 1 to 7 is shown as Table 10. For each dataset

\(^{6}\) This AUC is calculated by computing the definite integral using the trapezoidal
Figure 4. Average classification rates for the WebKB Texas dataset.

Figure 5. Average classification rates for the WebKB Washington dataset.

(i.e., for each row in the table), the best result among all kernels appears in bold. In addition, if the best performing method is not a kernel, the best result among all methods also appears in bold. The last two rows show, for each method, the AUC score averaged on all datasets as well as the ranking of kernel methods.

$K_{RCT}$ and $K_{RL}$ clearly outperform the other kernels on every dataset, with a small advantage for $K_{RCT}$. More precisely, $K_{RCT}$ produces the best results on almost every dataset except for CORA, while $K_{RL}$ obtains the best results on CORA and comes second (after $K_{RCT}$) on the other datasets. They appear

approximation technique (Stoer & Bulirsch, 2002; Dahlquist & Bjorck, 1974).
Figure 6. Average classification rates for the WebKB Wisconsin datasets.

Figure 7. Average classification rates for the Newsgroup dataset.

to be very stable across datasets (as also pointed out in collaborative recommendation). Both are competitive with the $D$-walks approach, except for the WebKB-Cornell dataset which looks somewhat atypical. Both $K_{RWR}$ and $K_{MD}$ are competitive but never get better results than the two leading kernels.

Some other kernels perform less well. $K_{CT}$, which provides rather good results on collaborative recommendation, lies behind on almost every dataset, except for the Newsgroup one. $K_{LED}$ gets decent results on the CORA, IMDb, and Newsgroup datasets, but performs badly on the four WebKB datasets, making it unstable to use in other situations. This observation is in accordance with the results provided by $K_{LED}$ on collaborative recommendation. Other kernels, such as $K_{ED}$, $K_{VND}$, and $K_{RED}$ provide poor and unstable results on almost
every dataset, confirming their poor results on collaborative recommendation.

We observe that the best kernel-based methods, while competitive in most cases, do not perform better than the semisupervised $D$-walks approach. This indicates that kernel methods appear to be useful general-purpose tools that deserve investigation, but also that they do not outperform domain-specific techniques, at least on the explored datasets.

<table>
<thead>
<tr>
<th></th>
<th>KED</th>
<th>KLED</th>
<th>KVND</th>
<th>KRL</th>
<th>KRCT</th>
<th>KCT</th>
<th>KMD</th>
<th>KRED</th>
<th>KRWR</th>
<th>D-walks</th>
<th>NetKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cora</td>
<td>0.100</td>
<td>0.629</td>
<td>0.266</td>
<td>0.477</td>
<td>0.503</td>
<td>0.674</td>
<td>0.209</td>
<td>0.671</td>
<td>0.680</td>
<td>0.597</td>
<td></td>
</tr>
<tr>
<td>imDb</td>
<td>0.401</td>
<td>0.629</td>
<td>0.482</td>
<td>0.647</td>
<td>0.648</td>
<td>0.633</td>
<td>0.647</td>
<td>0.615</td>
<td>0.645</td>
<td>0.559</td>
<td></td>
</tr>
<tr>
<td>WebKB-Cornell</td>
<td>0.331</td>
<td>0.331</td>
<td>0.266</td>
<td>0.477</td>
<td>0.503</td>
<td>0.402</td>
<td>0.462</td>
<td>0.010</td>
<td>0.470</td>
<td>0.672</td>
<td>0.436</td>
</tr>
<tr>
<td>WebKB-Washington</td>
<td>0.385</td>
<td>0.388</td>
<td>0.407</td>
<td>0.581</td>
<td>0.617</td>
<td>0.538</td>
<td>0.567</td>
<td>0.030</td>
<td>0.578</td>
<td>0.624</td>
<td>0.536</td>
</tr>
<tr>
<td>WebKB-Texas</td>
<td>0.310</td>
<td>0.311</td>
<td>0.420</td>
<td>0.537</td>
<td>0.557</td>
<td>0.347</td>
<td>0.548</td>
<td>0.010</td>
<td>0.517</td>
<td>0.550</td>
<td>0.490</td>
</tr>
<tr>
<td>NewsGroup</td>
<td>0.190</td>
<td>0.351</td>
<td>0.532</td>
<td>0.593</td>
<td>0.617</td>
<td>0.582</td>
<td>0.583</td>
<td>0.030</td>
<td>0.590</td>
<td>0.621</td>
<td>0.557</td>
</tr>
<tr>
<td>Average AUC</td>
<td>0.268</td>
<td>0.471</td>
<td>0.473</td>
<td>0.691</td>
<td>0.616</td>
<td>0.544</td>
<td>0.595</td>
<td>0.216</td>
<td>0.594</td>
<td>0.640</td>
<td>0.548</td>
</tr>
<tr>
<td>Ranking</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>/</td>
<td>/</td>
<td></td>
</tr>
</tbody>
</table>

Table 10

*Area Under Curve* score for each method and each dataset.

Similar to Table 5 for collaborative recommendation, Table 11 shows a summary of the scores obtained on the semisupervised classification experiments by the various kernel-based algorithms (see Section 5.5 for details about computing scores). Each row corresponds to one dataset, for a fixed labeling rate of 30%. The “baseline score” is the score obtained by NetKit. The “Average SC score” row computes, for each scoring algorithm, the average of its 7 results, with the same weight for all the criteria (i.e., the rows). The “SC ranking” row ranks the algorithms according to their average score. Table 11 shows that the best kernel-based algorithm (in terms of its average SC score) is $K_{RCT}$ (0.74), followed by $K_{RL}$ (0.65), and by $K_{MD}$ (0.64). A paired $t$-test shows that there is a significant difference (with a $p$-value smaller than 0.05) between the results provided by the best algorithm (i.e., $K_{RCT}$) and the results provided by the other ones. Two more algorithms provide scores that are above average (i.e., with a positive average SC score): $K_{RWR}$ (0.61) and $K_{CT}$ (0.31), while the last four algorithms provide scores which are below average: $K_{ED}$ (−1.28), $K_{LED}$ (−0.06), $K_{VND}$ (−0.17), and $K_{RED}$ (−1.44).

Note that the rankings based on AUC scores, shown in Table 10, are clearly confirmed by those based on overall scores in Table 11 (except for $K_{LED}$ and $K_{VND}$ whose rankings – i.e., 6 and 7 – are reversed).

<table>
<thead>
<tr>
<th></th>
<th>KED</th>
<th>KLED</th>
<th>KVND</th>
<th>KRL</th>
<th>KRCT</th>
<th>KCT</th>
<th>KMD</th>
<th>KRED</th>
<th>KRWR</th>
<th>D-walks</th>
<th>NetKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMDB</td>
<td>-2.17</td>
<td>-0.39</td>
<td>-1.24</td>
<td>0.58</td>
<td>0.58</td>
<td>0.45</td>
<td>0.58</td>
<td>0.26</td>
<td>0.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cora</td>
<td>-1.97</td>
<td>-0.61</td>
<td>0.15</td>
<td>0.61</td>
<td>0.58</td>
<td>0.39</td>
<td>0.58</td>
<td>-1.49</td>
<td>0.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebKB-Cornell</td>
<td>-0.13</td>
<td>-0.13</td>
<td>-0.55</td>
<td>0.75</td>
<td>0.94</td>
<td>0.05</td>
<td>0.67</td>
<td>-2.30</td>
<td>0.70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebKB-Texas</td>
<td>-0.36</td>
<td>-0.36</td>
<td>-0.22</td>
<td>0.69</td>
<td>0.87</td>
<td>0.41</td>
<td>0.64</td>
<td>-2.33</td>
<td>0.67</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebKB-Washington</td>
<td>-0.49</td>
<td>-0.49</td>
<td>0.17</td>
<td>0.77</td>
<td>0.91</td>
<td>-0.24</td>
<td>0.88</td>
<td>-2.19</td>
<td>0.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WebKB-Wisconsin</td>
<td>-1.23</td>
<td>-0.45</td>
<td>0.38</td>
<td>0.67</td>
<td>0.75</td>
<td>0.59</td>
<td>0.64</td>
<td>-1.99</td>
<td>0.64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NewsGroup</td>
<td>-2.09</td>
<td>-0.92</td>
<td>0.14</td>
<td>0.50</td>
<td>0.54</td>
<td>0.50</td>
<td>0.50</td>
<td>-0.92</td>
<td>0.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average SC score</td>
<td>-1.28</td>
<td>-0.86</td>
<td>-0.17</td>
<td>0.65</td>
<td>0.74</td>
<td>0.31</td>
<td>0.64</td>
<td>-1.44</td>
<td>0.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SC ranking</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11

Overall scores for ranking the algorithms on the semisupervised-classification (SC) task (the higher the value, the better the algorithm).
Table 12

<table>
<thead>
<tr>
<th>KRD</th>
<th>KLED</th>
<th>KVMN</th>
<th>KRL</th>
<th>KRCT</th>
<th>KCT</th>
<th>KMD</th>
<th>KRED</th>
<th>KbWb</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average CR score</td>
<td>-0.96</td>
<td>-0.62</td>
<td>-1.30</td>
<td>0.54</td>
<td>0.77</td>
<td>0.70</td>
<td>0.97</td>
<td>0.28</td>
</tr>
<tr>
<td>CR ranking</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Average SC score</td>
<td>-1.28</td>
<td>-0.06</td>
<td>-0.17</td>
<td>0.65</td>
<td>0.74</td>
<td>0.31</td>
<td>0.64</td>
<td>-1.44</td>
</tr>
<tr>
<td>SC ranking</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Average score</td>
<td>-1.12</td>
<td>-0.34</td>
<td>-0.74</td>
<td>0.60</td>
<td>0.76</td>
<td>0.51</td>
<td>0.81</td>
<td>-0.58</td>
</tr>
<tr>
<td>Final ranking</td>
<td>9</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 12 Overall scores for ranking the algorithms on collaborative recommendation (CR) and semisupervised classification (SC) (the higher the value, the better the algorithm).

7 General discussion

Table 12 shows scores obtained by combining Table 5 and Table 11. More precisely, the “Average score” is computed by averaging “Average CR” and “Average SC” scores (thus giving the same weight to both tasks). A “Final ranking” based on the “Average score” is assigned to each kernel. These results suggest that KMD is the best method, with an average score of 0.81, closely followed by KRCT (with a score of 0.76), KRL (with a score of 0.60), and KCT (with a score of 0.51). Still, (1) we cannot conclude that KMD provides significantly better results than KRCT (using a paired t-test to determine if there is a significant difference between the algorithms, p < 0.05) and (2) the results of Table 12 confirm that the choice of the best kernel is application-dependent (KMD for collaborative recommendation and KRCT for semisupervised classification); the most application-independent method (i.e., with the closest CR and SC scores) is KRCT (0.77 and 0.74).

Three kernels provide quite stable and good overall performance on all datasets, independently of the application, namely KMD, KRCT, and KRL. All the other kernels perform significantly worse on at least one task in our experiments. However, these best kernel-based methods, while competitive, do not always perform better than more domain-specific techniques, at least for the investigated datasets. This indicates that both the choice of kernels as well as their performance in comparison to more domain-specific techniques are to a certain extend domain-, or even data-, dependent.

8 Conclusion and further work

This paper investigates several kernel-based procedures to compute similarities between elements of a database or, more generally, nodes of an undirected weighted graph. These similarity measures can be used to compare items belonging to database tables that are not necessarily directly connected. They depend on the strength of the connectivity between items. Our experiments show that some kernels perform well in comparison with standard algorithms, namely the Markov diffusion kernel, the regularized Laplacian kernel, and the regularized commute-time kernel. These three kernels perform consistently well. They have a nice and intuitive interpretation in terms of random walk (the Markov diffusion kernel and the regularized commute-time kernel) or in terms of the matrix-forest theorem (the regularized Laplacian kernel).
Another interesting property of the kernel-based methods is that they induce a data matrix on which standard multivariate statistical analysis methods can be applied for mining graphs or databases and that this data matrix need not be computed (the kernel matrix is sufficient). This allows to generalize useful techniques, like clustering, principal-component analysis, discriminant analysis, and canonical correlation analysis to the study of graphs or databases.

Further work will investigate the interest of adding more features of users and items, and ratings on the edges of the adjacency matrix. Other kernel-based multivariate statistical techniques, like discriminant analysis or canonical correlation analysis, will also be studied in the context of graph mining. Finally, various approximation schemes allowing to perform semisupervised classification on large graphs will be studied.

References


Ding, C., Jin, R., Li, T., & Simon, H. (2007). A learning framework using Green’s function and kernel regularization with application to recom-


to link analysis. *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 586–592.


