communication refusée

UNIVERSITÉ LIBRE DE BRUXELLES
FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

Novel Measures on Directed Graphs and Applications to Large-Scale Within-Network Classification*

Thèse présentée par

Amin Mantrach

En vue de l'obtention du grade de

Docteur en Sciences

Septembre 2010

*Cette version, non-définitive, est remise en dépôt pour une défense privée et pourra faire objet de modifications suite aux remarques du jury. La version finale sera remise lors de la défense publique.

29/09/10

Preface

This preface will be completed for the final version that will be presented at the public defense.

Abstract

In recent years, networks have become a major data source in various fields ranging from social sciences to mathematical and physical sciences. Moreover, the size of available networks has grow substantially as well. This has brought with it a number of new challenges, like the need for precise and intuitive measures to characterize and analyze large scale networks in a reasonable time.

The first part of this thesis introduces a novel measure between two nodes of a weighted directed graph: The **sum-over-paths covariance**. It has a clear and intuitive interpretation: *two nodes are considered as highly correlated if they often co-occur on the same – preferably short – paths.* This measure depends on a probability distribution over the (usually infinite) countable set of paths through the graph which is obtained by minimizing the total expected cost between all pairs of nodes while fixing the total relative entropy spread in the graph. The entropy parameter allows to bias the probability distribution over a wide spectrum: going from natural random walks (where all paths are equiprobable) to walks biased towards shortest-paths. This measure is then applied to semi-supervised classification problems on medium-size networks and compared to state-of-the-art techniques.

The second part introduces three novel algorithms for within-network classification in large-scale networks, i.e., classification of nodes in partially labeled graphs. The algorithms have a **linear computing time in the number of edges, classes and steps** and hence can be applied to large scale networks. They obtained competitive results in comparison to state-of-the-art technics on the large scale `U.S. patents` citation network and on eight other data sets. Furthermore, during the thesis, we collected a novel benchmark data set: the **`U.S. patents` citation network**. This data set is now available to the community for benchmarks purposes.

The final part of the thesis concerns the combination of a citation graph with information on its nodes. We show that **citation-based** data provide better results for classification than **content-based** data. We also show empirically that combining both sources of information (content-based and citation-based) should be considered when facing a text categorization problem. For instance, while classifying journal papers, considering to extract an external citation graph may considerably boost the performance.

However, in another context, when we have to directly classify the network citation nodes, then the help of features on nodes will not improve the results.
The theory, algorithms and applications presented in this thesis provide interesting perspectives in various fields.

Contents

Chapter 1

Introduction

1.1 Context

Network and link analysis are important growing fields that are the subject of much recent work in various areas of science: applied mathematics, computer science, social science, physics, pattern recognition, applied statistics, to name a few. Within this context, one key issue is the proper definition of a similarity measure between the nodes of a graph, capturing their relationship and taking both direct and indirect links into account. Indeed, the input graph is typically very sparse, each node being connected to only a few neighbors. Therefore, the goal is to go from "local" to "global", from the local neighborhood inputs to the definition of a global similarity taking into account all nodes of the graph, considering indirect links as well.

As a first contribution, this work introduces a link-based **covariance measure between the nodes of a weighted directed graph**. To this end, a probability distribution over the (usually infinite) countable set of paths through the graph is defined by minimizing the total expected cost between all pairs of nodes while fixing the total relative entropy spread in the graph. This results in a Boltzmann distribution on the set of paths such that long (high-cost) paths occur with a low probability while short (low-cost) paths occur with a high probability. A sum-over-paths (SoP) covariance measure between nodes is then defined according to this probability distribution which has a clear, intuitive, interpretation: *two nodes are considered as highly correlated if they often co-occur on the same – preferably short – paths*. The resulting covariance matrix between nodes is a Gram matrix and therefore defines a valid kernel on the graph. It is obtained by inverting a $n \times n$ matrix, where n is the number of nodes in the graph, depending on the costs assigned to the arcs. In the same spirit, a **betweenness** score is also defined, measuring *the expected number of times a node occurs on a path*. As will be shown, the proposed measures can be used for various graph mining tasks such as computing betweenness centrality, semi-supervised classification of nodes, visualization, etc.

Many graph-based measures (e.g. betweenness, relatedness, kernels on graph) generally requires the order of $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ memory which is prohibitive on large-scale networks. This thesis investigates how graph-based measures, like the SoP covariance, may be exploited on large-scale directed sparse networks. In this spirit, this work addresses graph-based semi-supervised classification and betweenness computation in large, sparse, networks (several millions of nodes). The objective of semi-supervised classification is to assign labels to unlabeled nodes using the whole topology of the graph and the labeling at our disposal. Two approaches are developed to *avoid explicit computation of pairwise proximity* between the nodes of the graph, which would be impractical for graphs containing millions of nodes. The first approach directly computes, for each class, the sum of the similarities between the nodes to be classified and the labeled nodes of the class as initially suggested by Zhou et al. (2003, 2005). Along this approach, two algorithms exploiting different state-of-the-art kernels on a graph are developed. The same strategy can also be used in order to compute a betweenness measure. The second approach works on a lattice structure built from biased random walks on the graph, extending an idea introduced by Callut et al. (2008). These random walks allow to define a biased bounded betweenness for the nodes of interest, defined separately for each class. All the proposed algorithms have a **linear computing time** in the number of edges, and hence are *applicable to large sparse networks*. They are empirically validated on medium-size standard data sets and are shown to be competitive with state-of-the-art techniques. We propose a significative contribution to the machine learning community: an original huge data set, the **U.S. patents citation network**. This network is made up of more than 3 millions of granted patents that are linked through cited-citing relations (38M edges) distributed among six classes. This data set is made available to the community for benchmark purposes. The three proposed algorithms achieve competitive results (around 85% classification rate) on this large network – they classify the unlabeled nodes within a few minutes on a standard workstation.

The final part of this thesis is dedicated to the problem of combining graph-based measures (e.g. measures that takes into account only the graph topology) with other source of information (e.g. features on nodes). First, it shows that **citation-based** data provide better results for classification than **content-based** data. Secondly, it proposes different ways to combine topological information linking the data to the available information on each node. Indeed, standard machine learning approaches make the assumption that the data are independently and identically distributed which breaks any possibility of interconnection between the data. Therefore, we propose to apply graph-based semi-supervised learning approaches to other sources of information, i.e., not only the citation graph. We show empirically that combining both sources of information (content-based and citation-based) should be considered when facing a text categorization problem. For instance, while classifying journal papers, considering to extract an

external citation graph constitute an important issue. In this case, the documents you have to classify may benefit of the labels propagation coming from the citation graph. However, in another context, when we have to classify citation nodes, then the help of features on nodes (e.g. text abstracts in the case of papers) will not improve the results.

1.2 Main Contributions of the Thesis

The main contributions of this thesis can be summarized as follows:

- It introduces well-founded **covariance** and **betweenness** measures between the nodes of weighted directed graphs. The resulting covariance matrix defines a valid kernel on the graph.
- It shows how these covariance and betweenness measures can be computed efficiently from the immediate costs associated to each arc by inverting a $n \times n$ matrix, where n is the number of nodes in the graph.
- Experimental comparisons with several graph kernels, computed on eight different databases, show that the sum-over-paths (SoP) correlation measure obtains competitive performances in semi-supervised classification tasks.
- It provides three algorithms to address within-network semi-supervised classification tasks on large, sparse, directed graphs. All these algorithms have a **computing time linear in the number of edges** of the graph.
- It validates the three proposed algorithms on eight medium-size standard data sets and compares them to state-of-the-art techniques. Their performances are shown to be competitive in comparison with state-of-the-art techniques.
- It provides algorithm to compute betweenness centrality on large scale network: the SoP betweenness centrality and the bDWALK betweenness.
- It introduces a **novel benchmark data set** to the community: the `U.S. patents` citation network, on which our three algorithms obtain competitive results.
- It shows empirically, through systematic experiments on seven selected data sets, that connecting documents to an external citation graph resource can significantly improve the accuracy for a semi-supervised classification task (i.e. 10 to 30% increase of the classification rate is achieved depending on the data set).
- It shows empirically, through systematic experiments on seven selected data sets, that better classification accuracy are obtained with graph-based semi-supervised methods applied to citation-based graphs than to content-based graphs.
- Finally, it shows, that combining citation-based graphs to content-based graphs does not give significantly better accuracy results.

1.3 Organization of the Thesis

Chapter 2 recalls basic concepts related to this thesis. More precisely, basic notions about graph theory is given in Section 2.1, a brief survey on markov chains in Section 2.2 and kernel matrices are introduced in Section 2.3.

Chapter 3 introduces the randomized-shortest path framework of Saerens et al. (2009).

Chapter 4 is dedicated to similarity measures on graphs. It first introduces state-of-the-art centrality and relatedness measures on graphs. Then, a novel well-founded **covariance** as well as a novel **betweenness** measure between nodes of a weighted directed graph are defined.

Chapter 5 considers the problem of within-network classification (i.e. graph-based semi-supervised learning). Starting from basic assumptions, quadratic criterions are defined on which a regularization framework is discussed (see, e.g. Zhou and Scholkopf, 2004; Zhou et al., 2005). Different well-known kernels on graphs are derived from this regularization framework. This corresponds to a sum-of-similarities framework which is then used for a semi-supervised task in Section 5.3.

Chapter 6 investigates the problem of within-network classification on large scale graphs. In Section 6.1.3 a linear computing time algorithm for approximating the SoP covariance in a semi-supervised classification task is proposed. Section 6.1.4 proposes yet another linear-computing algorithm based on the random-walk with restart paradigm used by Page et al. (1999) for their famous pagerank algorithm. In Section 6.2, a third linear-time, random-walk-like, algorithm, is proposed working on a time unfolding of the original graph (i.e. a lattice). The **U.S. patents network collected during this thesis** is fully described in Section 6.3.2. Experimental results for within-classification on this `U.S. patents` data set are then discussed in the same Section.

Chapter 7 investigates how to combine node features to an already existing graph. Section 7.1 introduces different ways to build a graph from observations. Section 7.2 compares graph-based semi-supervised methods to the SVM state-of-the-art text categorization method. Section 7.3 builds k nearest neighbors graphs from different data sets and apply different graph-based semi-supervised algorithms on them. Section 7.4 proposes two different experiments by combining citation graphs to their content-based graphs.

Publications by the author

The present thesis is the result a collective work. However, the participation of the thesis author is not significant in all publications and only a part of the articles contribute to this work.

Journal Papers

Amin Mantrach, Luh Yen, Jerome Callut, Kevin Francoisse, Masashi Shimbo, and Marco Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1112Ð1126, 2010. ISSN 0162-8828
This work is introduced in

Submitted papers and Manuscripts in Preparation

Amin Mantrach, Nicolas van Zeebroeck, Pascal Francq, Masashi Shimbo, Hugues Bersini and Marco Saerens. Semi-supervised Classification and Betweenness Computation on Large, Sparse, Directed Graphs, *submitted for publication to Pattern Recognition*, PR-D-09-01097R. Minor Revision.
Caroline Herssens, **Amin Mantrach** and Marco Saerens, Ant colony optimization revisited from a randomized shortest path perspective, submitted for publication

International Conference Papers

L. Yen, **A. Mantrach**, M. Shimbo, and M. Saerens. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. *Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785Ð793, 2008.
L. Kevers, **A. Mantrach**, C. Fairon, H. Bersini and M. Saerens (2010), Classification supervisée hybride par motifs lexicaux étendus et classificateurs SVM, *10th International Conference on statistical analysis of textual data (JADT 2010)*, Rome, 9-11/06/2010, S. Bolasco, I. Chiari, L. Giuliano ed(s), Ed. Univ. di Lettere Economia Diritto, 2010, p. 105-117.

International WorkShops

Amin Mantrach and Marco Saerens, The All-Paths Covariance: a new covariance measure between nodes of a weighted, directed, graph, *MLG 2008 - 6th International Workshop on Mining and Learning with Graphs*, ID 15.

Chapter 2

Preliminaries

This thesis aims to study nodes involved in a network and more precisely to make use of the relations among them. Therefore we first propose, in Section 2.1, a brief introduction to graphs in order to deliver to the reader the minimum requirements about graph concepts: nodes, links, paths, connectivity, etc. What is also important to study is how we can walk around in the network. For example, when surfing on the web we would like to know what is the likelihood to click on a specific hyperlink? Or on a road network, what probability has the driver to follow a specific path? Markov chains by introducing graph-based probability models can answer to this kind of questions. In this thesis, we will also use meaningful similarity measures defined on every pair of nodes in a graph. Defining similarity measures is important for various applications: classification, visualization, clustering, etc. Mathematically, most of these similarity matrices are graph kernels, i.e., positive semidefinite matrices. Hence, a simple introduction of the basic notions of kernels is provided in Section 2.3.

2.1 Graphs and Networks

Graphs are mathematical structures used to model pairwise relations between different objects (for references on the subject see, e.g. Bollobas, 1998; Chung, 1997; West et al., 2001; Jungnickel, 2005). In this context, a graph is a set of nodes and a set of links connecting each pair of nodes. Graphs are often used to study different kinds of networks. For example, in social science, social networks are studied via graphs. In this context, nodes represent social entities (i.e. actors) and links indicate the social relationship between these entities. Modeling networks as a graph gives the opportunity to apply the various metrics and algorithms developed in graph theory to infer some properties on the original network.

2.1.1 Basic Definitions

A graph $\mathcal{G}$ is an ordered pair (V, E), where V is a set of *vertices* or *nodes* and E is a set of *edges* or *links*. When there is no specific orientation of the edges the graph is said to be *undirected*. When the edges are *directed* from node i to node j then the graph is *directed*. In this case, the directed edges are referred as arcs. A directed graph $\mathcal{D}$ is said to be *symmetric* if, for every directed edge in $\mathcal{D}$, the inverted edge belongs also to $\mathcal{D}$. A graph is weighted if a weight is assigned to each edge. Such weight might represent costs, distance, affinity, etc. Two edges of a graph are adjacent if they share a common vertex. To represent which vertices of a graph are adjacent to which other we commonly use an *adjacency matrix*. The adjacency matrix $\mathbf{A}$ of a finite graph $\mathcal{G}$ of n nodes, is a $n \times n$ matrix where the entry a_{ij} is the weight on the directed edge linking node i to node j. If the graph is undirected, then the adjacency matrix is *symmetric*.

2.1.2 Paths and Connectivity

In a graph, a *path* is a sequence of nodes such that from each node there exists an edge that links to the next node in the sequence. In this work, we consider that a *path* (sometimes also called "*walk*") may contain cycles (i.e. nodes may be repeated in the path sequence). In an undirected graph $\mathcal{G}$, two nodes i and j are *connected* if there exists a path from i to j otherwise they are *disconnected*. A graph is said to be *connected* if every pair of nodes in the graph may be connected through some existing path. The maximal connected subgraphs of $\mathcal{G}$ form the *connected components* of this graph. A directed graph is said to be *weakly connected* if replacing all of its directed edges with undirected edges produces a connected undirected graph. It is *connected* if, for every pair of nodes (i, j), it contains either a directed path going from node i to node j or a directed path going from node j to node i. And it is *strongly connected* if is contains both. The maximal strongly connected subgraphs form the *strongly connected components* of the graph.

2.2 Markov Chains Fundamentals

A discrete time markov chain (MC) (for more details see, e.g., Doyle and Snell, 1984; Kemeny and Snell, 1976; Norris, 1997) is a stochastic process $X_t | t \in \mathbb{N}$ where the random variable X takes its value at any discrete time t in a countable set W

$$\mathrm{P}(X_t = w | X_0, \ldots, X_{t-1}) = \mathrm{P}(X_t = w | X_{t-p}, \ldots, X_{t-1}) \tag{2.1}$$

and p is then the model order. If W is a finite alphabet Σ of discrete events $s_1, s_2, \ldots$ then the Markov chain is finite. When the order of the model is 1 then the dependence

is restricted to the previous event. A Markov chain is stationary if $p_{ij} = \mathrm{P}(X_t = s_j | X_{t-1} = s_i)$ does not depend on t.

Hence, a stationary finite order 1 MC is characterized by a *transition matrix* $\mathbf{P}$ and an *initial probability vector* π^0 with

$$\pi_i^0 = \mathrm{P}(X_0 = s_i) \tag{2.2}$$

For example, the stationary finite order 1 Markov chain on Figure 2.1 is characterized by the alphabet $\Sigma = \{a, b\}$, by the transition probability matrix

$$\mathbf{P} = \begin{bmatrix} 0 & 1 \\ 0.6 & 0.4 \end{bmatrix}$$

and by the initial probability vector $\pi^0 = [0.2 \;\; 0.8]$

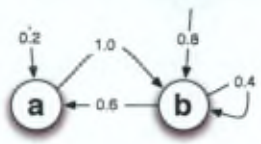


Figure 2.1: **Stationary order 1 finite state Markov chain.**

The likelihood $\mathrm{P}(s|M)$ of a sequence $s = s_{0 \cdots} s_{|s|-1}$ according to a MC M is

$$\mathrm{P}(s|M) = \prod_{i=0}^{|s|-1} \mathrm{P}(s_i|M) = \mathrm{P}(X_0 = s_0) \prod_{i=1}^{|s|-1} \mathrm{P}(X_i = s_i | X_{i-1} = s_{i-1}) \tag{2.3}$$

Working with the same Markov chain (Figure 2.1), we can compute the likelihood of the sequence: abba. $\mathrm{P}(\text{abba}) = \pi_a^0 \, p_{ab} \, p_{bb} \, p_{ba} = 0.2 \times 1 \times 0.4 \times 0.6$

We can compute the probability to be in a specific state i (each element of the alphabet s_i is mapped to a unique state i) after n steps : $\mathrm{P}(X_n = i)$

$$\mathrm{P}(X_n = i) = \sum_j \mathrm{P}(X_{n-1} = j) \mathrm{P}(X_n = i | X_{n-1} = j) \tag{2.4}$$

In matrix form we have

$$(\pi^n)^{\mathrm{T}} = (\pi^{n-1})^{\mathrm{T}} \mathbf{P} \tag{2.5}$$

Unfolding the recurrence we obtain

$$(\pi^n)^{\mathrm{T}} = (\pi^{n-1})^{\mathrm{T}} \mathbf{P} = (\pi^{n-2})^{\mathrm{T}} \mathbf{P}^2 = \cdots = (\pi^0)^{\mathrm{T}} \mathbf{P}^n \tag{2.6}$$

A related result is

$$\mathrm{P}(X_n = i | X_0 = j) = \mathrm{P}(X_{n+k} = i | X_k = j) = [\mathbf{P}^n]_{ij} \tag{2.7}$$

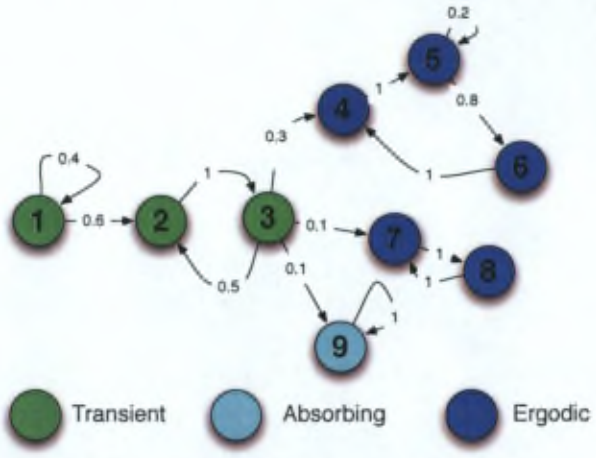


Figure 2.2: **Markov chain with different types of states.**

MC states are traditionally subdivided into three classes (Figure 2.2):

- *Ergodic sets*: This is a final set which is never left once entered.
- *Absorbing states*: A state is absorbing if it is the unique element of an ergodic set.
- *Transient states*: A non-ergodic set contains transient states. It will be denoted by $\mathcal{T}$

The greatest common divisor of the number of possible steps after which the process may return to the state defines the state period. A state is periodic if its period ≥ 2, otherwise it is aperiodic. An aperiodic ergodic chain is called regular.

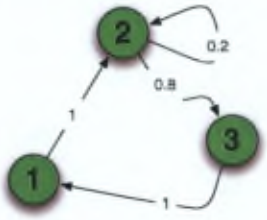


Figure 2.3: **Irreducible regular Markov chain.**

If the chain contains more than one ergodic set but no transient set then there is no interaction between them, i.e. they form fully separate chains. Therefore, one can

restrict his attention to irreducible MC which are formed of a single ergodic set (Figure 2.3).

In such a MC, we may want to compute the expected proportion of time being in a given state i. The solution is given by the unique stationary distribution of the irreducible MC

$$\vec{\pi}^{\mathrm{T}}\mathbf{P} = \vec{\pi}^{\mathrm{T}} \tag{2.8}$$

For any initial probability vector π^0

$$\lim_{n\to\infty} (\pi^0)^{\mathrm{T}}\mathbf{P}^n = \vec{\pi}^{\mathrm{T}} \tag{2.9}$$

This means that the process has the same probability to be in state i no matter from where it starts. In general, the probability transition matrix $\mathbf{P}$ of the MC may be arranged like

$$\mathbf{P} = \begin{bmatrix} \mathbf{Q} & \mathbf{R} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

where $\mathbf{Q}$ is the transition matrix between transient states, $\mathbf{R}$ is the transition matrix from the transient states to the absorbing states and $\mathbf{I}$ the identity transition matrix between absorbing states. We may want to compute the expected number of visits (passage times) on transient state j starting from i ($E(v_j|X_0 = i) = n_{ij}$). If i is an absorbing state then we never visit j. The interesting case is when both states i and j are transient.

$$n_{ij} = E[v_j|X_0 = i] = \delta_{ij} + \sum_{k\in T} p_{ik}E[v_j|X_0 = k] \tag{2.10}$$

$$n_{ij} = \delta_{ij} + \sum_{k\in T} q_{ik}n_{kj} \tag{2.11}$$

In matrix form

$$\mathbf{N} = \mathbf{I} + \mathbf{QN} \Rightarrow \mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1} \tag{2.12}$$

$\mathbf{N}$ is called the fundamental matrix of the MC.

For example, considering the Markov chain of Figure 2.4, the $\mathbf{Q}$ matrix has this form :

$$\mathbf{Q} = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 0 & 0 & 1 \\ 0 & 0.5 & 0 \end{bmatrix}$$

The associated fundamental matrix is then

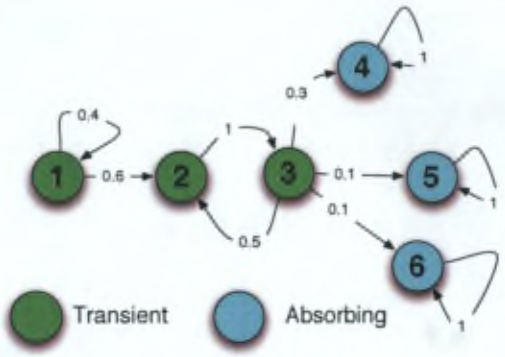


Figure 2.4: **Markov chain with one ergodic set (3 transient states) and 3 absorbing states.**

$$\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1} = \begin{bmatrix} \frac{5}{3} & 2 & 2 \\ 0 & 2 & 2 \\ 0 & 1 & 2 \end{bmatrix}$$

We can also compute the probability to reach node j starting from node i. We convert node j into an absorbing state (unless it was already absorbing) , and the other ergodic sets into single absorbing states. And we compute d_{ij} the probability of absorption in j starting from i. At the first step, the probability p_{ij} is simply the probability of transiting from transient state i to absorbing state j, i.e. r_{ij}. Otherwise, the process may have moved to another absorbing state such that there is in that case no additional chance to reach node j. And finally, the process may move to transient state k with probability q_{ik}. Then,

$$d_{ij} = r_{ij} + \sum_{k \in \mathcal{T}} q_{ik} d_{kj} \tag{2.13}$$

In matrix form, it gives

$$\mathbf{D} = \mathbf{R} + \mathbf{QD} \Rightarrow \mathbf{D} = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{R} = \mathbf{NR} \tag{2.14}$$

With the same example (Figure 2.4) the matrix $\mathbf{R}$ has the form

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.3 & 0.1 & 0.1 \end{bmatrix}$$

Matrix $\mathbf{D}$ can be computed then

With the same example (Figure 2.4) the matrix $\mathbf{R}$ has the form

$$\mathbf{D} = \mathbf{NR} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.6 & 0.2 & 0.2 \end{bmatrix}$$

When working on a graph, it may be seen as a MC. In this case, graph nodes are considered as states of the MC. A random walker is going on the graph from node to node following the probability transition matrix $\mathbf{P}$ of the underlying MC. The random variable $X(t)$ denotes the current state at time t. The transition probability is generally proportional to the edge weight

$$p_{ij} = \frac{a_{ij}}{\sum_k a_{ik}} \tag{2.15}$$

For a complete introduction to markov chains theory and random walks, see e.g., Doyle and Snell (1984); Kemeny and Snell (1976); Norris (1997). This brief introduction to Markov chain also takes its inspiration from Dupont (2007).

2.3 Kernels

As stated in Fouss et al. (2007b), a kernel matrix (for more details see, e.g., Scholkopf et al., 1998; Shawe-Taylor and Cristianini, 2004) is a matrix that is positive semidefinite, it is also called a Gram matrix. Any real symmetric matrix that is not positive semidefinite can be changed to a positive semidefinite one by adding the identity matrix $\mathbf{I}$ multiplied by a suitably positive value to the original matrix (for techniques converting similarity into semi positive definite matrices see, e.g., Chen and Ye, 2008; Chen et al., 2009; Roth et al., 2002; Rousseuw and Molenberghs, 1993; Wu et al., 2005). Indeed, if $\mathbf{M}$ is a symmetric matrix, the matrix $(\mathbf{M}+\alpha\mathbf{I})$, with $\alpha \geqslant |\lambda_{\min}|$ ($\lambda_{\min}$ is the smallest eigenvalue of $\mathbf{M}$) is positive semidefinite. This is because $\mathbf{M}$ and $(\mathbf{M}+\alpha\mathbf{I})$ have the same eigenvectors $\mathbf{u}_i$ associated with eigenvalues λ_i for $\mathbf{M}$ and $(\lambda_i+\alpha)$ for $(\mathbf{M}+\alpha\mathbf{I})$. Thus, by choosing $\alpha \geqslant |\lambda_{\min}|$, all the eigenvalues are non-negative and $(\mathbf{M}+\alpha\mathbf{I})$ is positive semidefinite.

The main concepts basically come from the fields of multidimensional scaling (see, e.g., Borg and Groenen, 1997; Cox and Cox, 2001; Mardia et al., 1979) and kernel methods (see, e.g., Scholkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004).

2.3.1 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, e.g., Mardia et al., 1979; Noble and Daniels, 1988): *Any $n \times n$ real positive semidefinite matrix $\mathbf{K}$ of rank r can be expressed as* $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T = \mathbf{U}\mathbf{\Lambda}^{1/2}(\mathbf{U}\mathbf{\Lambda}^{1/2})^T = \mathbf{X}\mathbf{X}^T$, *where $\mathbf{\Lambda}$ is the diagonal matrix containing the*

eigenvalues of $\mathbf{K}$, $\mathbf{U}$ *is the* $n \times r$ *orthonormal matrix whose columns are the normalized eigenvectors* $\mathbf{u}_i$ *of* $\mathbf{K}$, *and* $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}^{1/2}$ *is a* $n \times r$ *matrix of rank* r. *Thus, the* r *columns* $\mathbf{c}_i = \sqrt{\lambda_i}\mathbf{u}_i$ *of* $\mathbf{X}$ *correspond to the orthonormal eigenvectors* $\mathbf{u}_i$ *of* $\mathbf{K}$ *multiplied by the square root of their corresponding eigenvalue.*

Therefore, if $\mathbf{x}_i$ is the column vector corresponding to column i of $\mathbf{X}^\mathrm{T}$ (transposed row i of $\mathbf{X}$), then the entries of $\mathbf{K}$ are inner products $[\mathbf{K}]_{ij} = k_{ij} = \mathbf{x}_i^\mathrm{T}\mathbf{x}_j$, in accordance with the fact that $\mathbf{K}$ is a kernel. The nodes of the graph are thus represented as vectors $\mathbf{x}_i$ in a r-dimensional Euclidean space and they form a cloud of n points in $\Re^r$. The $\mathbf{x}_i$'s are called **node vectors**, while matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\mathrm{T}$ containing the transposed node vectors as rows is the **data matrix** associated to the graph kernel. 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 (see, e.g., Lee and Verleysen, 2007; Shawe-Taylor and Cristianini, 2004) but this question is not investigated in this thesis.

2.3.2 Centering the data matrix

Usually, there is a preference for working with centered vectors, so that the centre of gravity of the node vectors $\mathbf{x}_i$ is $\mathbf{0}$. For instance, most multivariate statistical techniques assume that the data has been centered on the center of gravity of the data cloud (see, e.g., Jolliffe, 2002; Mardia et al., 1979). This amounts to subtracting its mean to each feature of the data matrix, therefore normalizing it. A data matrix is centered if $\mathbf{X}^\mathrm{T}\mathbf{e} = \mathbf{0}$, that is, the sum of the elements of each column of $\mathbf{X}$ is 0. Now, it can easily be shown that a symmetric kernel matrix $\mathbf{K}$ is centered (that is, it corresponds to inner products of centered node vectors) if and only if $\mathbf{K}\mathbf{e} = \mathbf{0}$.

A centered kernel matrix $\overline{\mathbf{K}}$ is therefore defined by centering the node vectors, that is, by applying the symmetric centering matrix $\mathbf{H} = \mathbf{I} - \frac{\mathbf{e}\mathbf{e}^\mathrm{T}}{n}$ (see, e.g., Mardia et al., 1979) to the data matrix $\mathbf{X}$. Indeed, premultiplying a data matrix by $\mathbf{H}$ re-expresses each element of the matrix as a deviation from its column mean, i.e., $\overline{\mathbf{X}} = \mathbf{H}\mathbf{X}$ has its $(i,j)^{\text{th}}$ element $x_{ij} - \overline{x}_j$, where $\overline{x}_j$ is the mean of the j^{th} column of $\mathbf{X}$. Thus, the **centered kernel matrix** $\overline{\mathbf{K}}$ is defined as $\overline{\mathbf{K}} = (\mathbf{H}\mathbf{X})(\mathbf{H}\mathbf{X})^\mathrm{T} = \mathbf{H}\mathbf{X}\mathbf{X}^\mathrm{T}\mathbf{H} = \mathbf{H}\mathbf{K}\mathbf{H}$.

2.3.3 The cosine, or normalized, kernel matrix

Inner products are not always an appropriate similarity measure. In some fields, such as information retrieval (see, e.g., Baeza-Yates and Ribeiro-Neto, 1999; Manning et al.,

2008), the cosine similarity measure is preferred:

$$\cos_{ij} = \frac{\mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j}{||\mathbf{x}_i||||\mathbf{x}_j||} = \frac{k_{ij}}{\sqrt{k_{ii}k_{jj}}} \tag{2.16}$$

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

2.3.4 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:

$$\begin{aligned}\delta_{ij}^2 &= ||\mathbf{x}_i - \mathbf{x}_j||^2 = (\mathbf{x}_i - \mathbf{x}_j)^{\mathsf{T}}(\mathbf{x}_i - \mathbf{x}_j) \\ &= \mathbf{x}_i^{\mathsf{T}}\mathbf{x}_i + \mathbf{x}_j^{\mathsf{T}}\mathbf{x}_j - 2\mathbf{x}_i^{\mathsf{T}}\mathbf{x}_j = k_{ii} + k_{jj} - 2k_{ij} \\ &= (\mathbf{e}_i - \mathbf{e}_j)^{\mathsf{T}}\mathbf{K}(\mathbf{e}_i - \mathbf{e}_j)\end{aligned} \tag{2.17}$$

where $\mathbf{e}_i$ is a column vector with "0" entries, except in position i whose entry is "1" (i.e., $\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathsf{T}}$).

Since δ_{ij} corresponds to a Euclidean distance in $\mathbb{R}^r$, it satisfies all the properties of a distance (positiveness, triangular inequality, etc.). Distances between pairs of elements allow, for instance, to use a clustering algorithm to group the nodes that are most similar (see, e.g., Shawe-Taylor and Cristianini, 2004; Yen et al., 2007, 2005). In matrix form, $\mathbf{\Delta} = \mathbf{diag(K)e}^{\mathsf{T}} + \mathbf{e(diag(K))}^{\mathsf{T}} - 2\mathbf{K}$, where $\mathbf{diag(K)}$ is a column vector containing the diagonal elements of $\mathbf{K}$, and $[\mathbf{\Delta}]_{ij} = \delta_{ij}^2$ contains the squared distances (see, e.g., Borg and Groenen, 1997; Cox and 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 $\mathbf{\Delta}$, the natural centered kernel matrix associated to $\mathbf{\Delta}$ is $\mathbf{K} = -\frac{1}{2}\mathbf{H\Delta H}$ (see, e.g., Borg and Groenen, 1997; Cox and Cox, 2001; Mardia et al., 1979). Once a data matrix $\mathbf{X}$ has been derived from $\mathbf{K}$, 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 (see, e.g., Scholkopf et al., 1998; Shawe-Taylor and Cristianini, 2004). This brief introduction to kernel matrices also takes its inspiration from Fouss et al. (2007b).

Chapter 3

The Randomized Shortest Path Framework

In computer science, *shortest-paths* algorithms rank among widespread methods commonly used to solve problems. Many applications in machine learning, bioinformatics, data mining, speech recognition, natural language processing, link analysis, network analysis, reinforcement learning, routing etc. rely somehow on variants of *shortest-paths* algorithms. In routing, the process of selecting paths in a network along which to diffuse the traffic includes a *shortest-paths* detection procedure. Routing is performed for several kinds of networks: telecommunication network, data network (such as internet), transportation network, etc. To forward a packet, the routing procedure may use the distance routing protocol where the Bellman-Ford algorithm is used to identify shortest-paths (see, e.g., Bellman, 1958; Ford and Fulkerson, 1962). Link-state routing protocol may also be used where, this time, the Dijkstra algorithm solves the same problem of identifiying shortest-paths (see, e.g., Dijkstra, 1959). In sequence alignment, the best alignment is determined using the *shortest-path* leading the source sequence to the destination one using dynamic programming (see, e.g., Bertsekas, 2000). In bioinformatics a well-known dynamic programming algorithm, the Smith-Watterman algorithm, is commonly used for determining similar regions between two nucleotide or protein sequences (see, e.g., Smith and Watterman, 1981). In speech recognition, the Viterbi algorithm is used in hidden markov models to identify the best segmentation sequence - i.e. the *shortest path* - of a word according to phonemes. The speech recognition community also proposed the Baum-Welch algorithm, where this time all the paths are taken into account to learn the parameters of a hidden markov model by maximizing the likelihood of a set of sequences (see, e.g., Jelinek, 1997; Rabiner, 1989). In natural language processing the more likely parsing tree of a sentence is identified using dynamic programming (see, e.g., Jurafsky and Martin, 2000). In link

analysis and network analysis, centrality measures on nodes like the betweenness or the closeness rely on the counting of geodesic paths - i.e. *shortest paths* - in the network (see, e.g., Wasserman and Faust, 1994). All these algorithms share the common idea to (un)favour the *shortest-paths*.

Suppose, however, that you want to avoid a pure deterministic policy consisting of choosing always the *shortest paths*. Also, consider that you want to introduce some randomization in your process in order to be able to explore other paths than the shortest ones. Such a policy can be worthwhile if your environment is changing over time. Indeed, without exploration, new beneficial paths will never be discovered. Routing processes would benefit from new alternative paths. Indeed, alternative paths would allow the traffic to be diffused on multiple paths, hence reducing congestion effects. Furthermore, randomization can also be a beneficial alternative to the standards Viterbi and Baum-Welch algorithms. Indeed, exploration is able to identify other worthy paths than the shortest ones without necessarily taking all paths into account. In bioinformatics, protein and nucleotide sequences alignment algorithms would benefit from alternative paths between the source and the destination sequence. In network analysis, randomization would allow us to generalize centrality measures by taking into account alternative paths as well. In most cases, algorithms relying somehow on variants of shortest-paths algorithms may benefit of stochasticity.

In a shortest path setting, one can suppose to have a single source and want to find a single destination through a path of minimum length in the network, where the length is the sum of the costs of each transition on the path. In a deterministic routing policy we just follow the shortest-path at each step. Randomizing the routing policy implies that agents can follow different paths according to some probability distribution. So, instead of giving a probability 1.0 to shortest paths transitions, we accept to sacrifice the efficiency for exploration. Hence, transitions on alternative exploratory paths will have a non-zero probability, insuring exploration of the network.

In this stochastic shortest-path framework, we propose to measure the divergence associated with a given path in the network by the relative entropy (Kullback-Leibler) between the probability distribution on the set of paths in the network and the *natural* probability distribution associated with the network. The natural distribution probability, associated to the network, is a distribution that does not give any preference to a transition over another except on basis of the transition cost itself. So, on the one hand, if we fix the relative entropy to zero, it means that the two distributions are equal, which gives prominence to the natural walk on the network. In this case, the walk is completely random. On the other hand, increasing the relative entropy, i.e. decreasing the randomness, biases the walk against the natural random walk. This model, inspired by Akamatsu (1996) in the field of transportation science, has been formally studied by Saerens et al. (2009). This model is discussed in detail in the subsequent part of this chapter.

Organization of this Chapter

Section 3.1 introduces the necessary background and notations. Section 3.2 defines a Boltzmann distribution on the set of paths. Finally, Section 3.3 explains how to compute the associated partition function Z.

3.1 Background and notation

Consider a weighted directed graph or network, $\mathcal{G}$, not necessarily strongly connected, with a set of n nodes V (or vertices) and a set of arcs E (or edges). To each arc linking node k and k', we associate a positive number $c_{kk'} > 0$ representing the **immediate cost** of following this arc. The **cost matrix C** is the matrix containing the immediate costs $c_{kk'}$ as elements. A random walk on this graph is defined in the standard way. In node k, the random walker chooses the next arc to follow according to transition probabilities representing the probability of jumping from node k to node $k' \in S(k)$, the set of successor nodes (successors S). These transition probabilities will be denoted as $p_{kk'} = \mathrm{P}(X_t = k'|X_{t-1} = k)$ with $k' \in S(k)$. Furthermore, $\mathbf{P}$ will be the matrix containing the transition probabilities $p_{kk'}$ as elements. If there is no arc between k and k', we simply consider that $c_{kk'}$ takes a large value, denoted by ∞; in this case, the corresponding transition probability will be set to zero, $p_{kk'} = 0$.
The **natural random walk** on this graph will be defined in the following way. It corresponds to a standard random walk through the graph with transition probabilities that are fixed on basis of the graph. The corresponding transitions-probabilities matrix will be denoted as $\mathbf{P}^{\mathrm{ref}}$. It can be computed simply by using the out-degree of each node. For example,

$$p^{\mathrm{ref}}_{kk'} = \frac{1}{\text{out-degree of node } k} \tag{3.1}$$

Or, it can also be computed using the transition costs

$$p^{\mathrm{ref}}_{kk'} = \frac{1/c_{kk'}}{\sum_{k'}(1/c_{kk'})} \tag{3.2}$$

They are different way to choose and fix a reference distribution $\mathbf{P}^{\mathrm{ref}}$. In this work we prefer to choose as reference distribution the natural random walk one. In other words, the random walker chooses to follow an arc with a probability proportional to the inverse of the immediate cost (apart from the sum-to-one normalization), therefore locally favoring arcs having a low cost. These transition probabilities will be used as reference probabilities later; hence the superscript "ref". If, instead of $\mathbf{C}$, we are given an adjacency matrix with elements $a_{kk'} \geq 0$ indicating the affinity between node k and node k', the corresponding costs are computed from $c_{kk'} = 1/a_{kk'}$ and the transition probabilities associated to each node are simply proportional to the affinities

(and normalized). Notice that other relations – other than the inverse relation – between affinity and cost could be considered as well. The matrix $\mathbf{P}^{\text{ref}}$ is provided by the user, is assumed to be stochastic, constant and contains real, positive, numbers.

The objective of the next sections is to define the probability distribution on the set of paths. Before diving into the details, let us briefly describe the main ideas behind the model. In a first step, the potentially infinite set of paths in the graph is enumerated and a probability distribution is assigned to each individual path: the longer the path, the smaller the probability of following it. This probability distribution depends on a parameter, $\theta = 1/T$, controlling the exploration carried out in the graph: when θ is large, no exploration is performed and only the shortest paths are enumerated while when θ is small, a random walk on the graph is performed according to the natural transition probabilities $p^{\text{ref}}_{kk'}$.

3.2 A Boltzmann distribution on the set of paths

Following Saerens et al. (2009), the present section describes how the probability distribution on the set of paths is assigned. To this end, let us first choose two nodes, an initial node i and a destination node j and define the set of paths (including cycles) connecting these two nodes as $\mathcal{R}_{ij} = \{\wp_{r^{ij}}\}$. Thus, $\wp_{r^{ij}}$ is path number r^{ij}, with r^{ij} usually ranging from 1 to ∞. Let us denote as $E_{r^{ij}}$ the total cost associated to path number r^{ij}, referred to as the **energy** associated to that path. Here, we assume that $\wp_{r^{ij}}$ is a valid path from the initial node to the destination node, that is, every $c_{k_{t-1}k_t} \neq \infty$ along that path. In addition, let us define the set of all paths through the graph as $\mathcal{R} = \bigcup_{ij} \mathcal{R}_{ij}$. We further assume that the total cost associated to a path is additive, i.e. $E(\wp_{r^{ij}}) = \sum_{t=1}^{t_f} c_{k_{t-1}k_t}$ where $k_0 = i$ is the initial node and $k_{t_f} = j$ is the destination node while t_f is the time (number of steps) needed to end the path in node j. Now, a probability distribution on this set $\mathcal{R}$, representing the probabilities of following the paths $\wp \in \mathcal{R}$, is defined as the probability distribution P minimizing the total expected cost-to-go, $\mathrm{E}\{E(\wp)\}$, among all the distributions having a fixed relative entropy J_0 with respect to the natural random walk on the graph. This choice naturally defines a probability distribution on the set of paths such that long paths occur with a low probability while short paths occur with a high probability. In other words, we are seeking path probabilities, $\mathrm{P}(\wp), \wp \in \mathcal{R}$, minimizing the total expected cost subject to a constant relative entropy constraint:

$$\left| \begin{array}{ll} \underset{\mathrm{P}(\wp)}{\text{minimize}} & \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp) E(\wp) \\ \text{subject to} & \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp) \ln(\mathrm{P}(\wp)/\mathrm{P}^{\text{ref}}(\wp)) = J_0 \end{array} \right. \tag{3.3}$$

where $\mathrm{P}^{\mathrm{ref}}(\wp)$ represents the probability of following the path $\wp$ when walking according to the natural random walk, i.e. using transition probabilities $p^{\mathrm{ref}}_{kk'}$ (see Equation (3.2)). Here, $J_0 > 0$ is provided a priori by the user, according to the desired degree of randomness, i.e. relative entropy, he is willing to concede. By minimizing the Lagrange function over the set of path probabilities $\mathrm{P}(\wp)$

$$\begin{aligned}\mathcal{L} &= \sum_{\wp\in\mathcal{R}} \mathrm{P}(\wp)E(\wp) + \lambda\left[\sum_{\wp\in\mathcal{R}} \mathrm{P}(\wp)\ln\frac{\mathrm{P}(\wp)}{\mathrm{P}^{\mathrm{ref}}(\wp)} - J_0\right] \\ &\quad +\mu\left[\sum_{\wp\in\mathcal{R}} \mathrm{P}(\wp) - 1\right], \end{aligned} \tag{3.4}$$

we obtain a **Boltzmann probability distribution**:

$$\mathrm{P}(\wp) = \frac{\mathrm{P}^{\mathrm{ref}}(\wp)\exp\left[-\theta E(\wp)\right]}{\sum_{\wp\in\mathcal{R}} \mathrm{P}^{\mathrm{ref}}(\wp)\exp\left[-\theta E(\wp)\right]} \tag{3.5}$$

$$= \frac{\exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]}{\sum_{\wp\in\mathcal{R}} \exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]} \tag{3.6}$$

where the Lagrange parameter λ plays the role of a temperature ($\lambda = T$) and $\theta = 1/\lambda$ is the inverse temperature. Thus, as expected, short paths $\wp$ (having small $E(\wp)$) are favored in that they have a large probability of being followed. Indeed, from Equation (3.6), we clearly observe that when $\theta \to 0$, the paths probabilities reduce to the probabilities generated by the natural random walk on the graph (characterized by the transition probabilities $p^{\mathrm{ref}}_{kk'}$ as defined in Equation (3.2)). In this case, $J_0 \to 0$ as well. On the other hand, when θ is large, the first term in the exponential dominates the second one, so that the probability distribution defined by Equation (3.6) is biased towards short paths (the most likely paths are the shortest ones). Notice that, in the sequel, it will be assumed that the user provides the value of the parameter θ instead of J_0, with $\theta > 0$. Notice that the model could be derived thanks to a maximum entropy principle instead (see, e.g., Jaynes, 1957; Kapur and Kesavan, 1992).

As suggested by Saerens et al. (2009), the quantity, appearing in the denominator of Equation (3.6), is defined as

$$\mathcal{Z} = \sum_{\wp\in\mathcal{R}} \exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right], \tag{3.7}$$

and corresponds to the **partition function** in statistical physics (see Jaynes (1957) or any textbook in statistical physics; for instance Reichl (1998); Schrodinger (1952)).

For illustration, the graph in Figure 3.1 has one shortest path between node i and j,

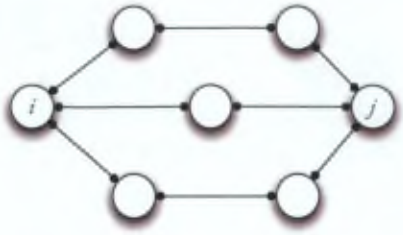


Figure 3.1: **A graph with one shortest-path between i and j. The edge weights are assumed to be 1.**

with an unitary cost on all transitions. Ignoring cycles and assuming uniform reference probabilities, the probability of the shortest-path is $\mathrm{P}(\wp) = \frac{1}{1+2\exp(-\theta)}$ while the probability of the two non-shortest paths is $\mathrm{P}(\wp) = \frac{\exp(-\theta)}{1+2\exp(-\theta)}$. For a large value of $\theta = 20$, the probability of the shortest-path is practically 1 while the other paths have an almost zero probability. However, for a low value $\theta \backsimeq 0$, the probability of all three paths going from node i to node j is almost equal to $1/3$. For an intermediary value of $\theta = 1$, the probability of the shortest-path is $\backsimeq 0.58$, which is higher than the probability of the non-shortest paths, $\backsimeq 0.21$.

3.3 Computation of the partition function $\mathcal{Z}$

By applying the ideas introduced by Akamatsu (1996), let us now show how the partition function $\mathcal{Z}$ (Equation (3.7)) can be computed exactly from the immediate costs. We start from the cost matrix, $\mathbf{C}$, from which we build a new matrix, $\mathbf{W}$, as

$$\mathbf{W} = \mathbf{P}^{\mathrm{ref}} \circ \exp\left[-\theta \mathbf{C}\right] = \exp\left[-\theta \mathbf{C} + \ln \mathbf{P}^{\mathrm{ref}}\right], \tag{3.8}$$

where $\mathbf{P}^{\mathrm{ref}}$ is the transition-probabilities matrix containing the $p^{\mathrm{ref}}_{kk'}$, and the logarithm/exponential functions are taken elementwise. Moreover, $\circ$ is the elementwise (Hadamard) matrix product.

Remember that $\wp_{r^{ij}}$ is path number r^{ij} between initial node i and destination node j. Now, since all the quantities in the exponential of Equation (3.7) are summed along a path, $\ln \mathrm{P}^{\mathrm{ref}}(\wp_{r^{ij}}) = \sum_{t=1}^{t_f} \ln p^{\mathrm{ref}}_{k_{t-1}k_t}$ and $E(\wp_{r^{ij}}) = \sum_{t=1}^{t_f} c_{k_{t-1}k_t}$ where each link $k_{t-1} \to k_t$ lies on path number r^{ij}, we easily observe that element (i,j) of the matrix $\mathbf{W}^t$ ($\mathbf{W}$ to the power t) is $[\mathbf{W}^t]_{ij} = \sum_{\wp_{r^{ij}} \in \wp^{ij}(t)} \exp[-\theta E(\wp_{r^{ij}}) + \ln \mathrm{P}^{\mathrm{ref}}(\wp_{r^{ij}})]$ where $\wp^{ij}(t)$ is the set of paths connecting the initial node i to the destination node j

in *exactly t steps*. Consequently, the partition function is

$$\mathcal{Z} = \sum_{\wp \in \mathcal{R}} \exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right] \tag{3.9}$$

$$= \sum_{i,j=1}^{n} \sum_{t=1}^{\infty} \sum_{\wp_{r^{ij}} \in \mathcal{P}^{ij}(t)} \exp\left[-\theta E(\wp_{r^{ij}}) + \ln \mathrm{P}^{\mathrm{ref}}(\wp_{r^{ij}})\right] \tag{3.10}$$

$$= \sum_{i,j=1}^{n} \left[\sum_{t=1}^{\infty} \mathbf{W}^t\right]_{ij} = \mathbf{e}^{\mathsf{T}} \left(\sum_{t=1}^{\infty} \mathbf{W}^t\right) \mathbf{e} \tag{3.11}$$

Thus, the sum over the elements of the matrix series $\sum_{t=1}^{\infty} \mathbf{W}^t$ corresponds to the partition function. Computing the series of powers of $\mathbf{W}$ provides

$$\sum_{t=1}^{\infty} \mathbf{W}^t = (\mathbf{I} - \mathbf{W})^{-1} - \mathbf{I} \tag{3.12}$$

which converges if the spectral radius of $\mathbf{W}$ is less than 1, $\rho(\mathbf{W}) < 1$. Since the matrix $\mathbf{W}$ only contains non-negative elements, a sufficient condition for $\rho(\mathbf{W}) < 1$ is that all its row sums are less than 1 (the matrix is sub-stochastic), which is always achieved for $\theta > 0$ since the $c_{kk'} > 0$ (see Equation (3.8)). Indeed, it is well-known that the spectral radius of a real square matrix is always smaller than or equal to its maximum absolute row sum norm (see, e.g., Bronson (1989), p. 111). Equation (3.12) is therefore well-defined provided $\theta > 0$.

Now, if we pose $\mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1}$, the partition function is

$$\mathcal{Z} = \mathbf{e}^{\mathsf{T}} \left((\mathbf{I} - \mathbf{W})^{-1} - \mathbf{I}\right) \mathbf{e} \tag{3.13}$$

$$= \mathbf{e}^{\mathsf{T}} (\mathbf{Z} - \mathbf{I}) \mathbf{e} \tag{3.14}$$

$$= \mathbf{e}^{\mathsf{T}} \mathbf{Z} \mathbf{e} - n \tag{3.15}$$

$$= z_{\bullet\bullet} - n \tag{3.16}$$

where $z_{\bullet\bullet} = \sum_{k,k'=1}^{n} z_{kk'}$ and $z_{kk'}$ is element k, k' of $\mathbf{Z}$. By analogy with Markov chains, $\mathbf{Z}$ will be called the **fundamental matrix**. An intuitive interpretation of the elements $z_{kk'}$ of the $\mathbf{Z}$ matrix can be given. Consider a special random walk defined by the transition-probabilities matrix $\mathbf{W}$. Since $\mathbf{W}$ is sub-stochastic, the random walker has a non-zero probability of disappearing at each node k and each time step which is equal to $(1 - \sum_{k'} w_{kk'})$. From Equation (3.8), it can be observed that the probability of surviving during a transition $k \to k'$ is proportional to $\exp[-\theta c_{kk'}]$, since $\mathbf{P}^{\mathrm{ref}}$ is a constant. This interpretation makes sense: there is a smaller probability to survive edges with a high cost. In this case, the elements of the $\mathbf{Z}$ matrix, $z_{kk'} = [\mathbf{Z}]_{kk'}$, can be interpreted as the expected number of passages through node k' (see for instance Doyle and Snell (1984); Kemeny and Snell (1976)) for an "evaporating" random walker

starting in node k.

Chapter 4

Novel Betweenness and Covariance Measures between Nodes of a Directed Graph

Determining the importance of a node (or a link*) within a graph is a big issue in graph theory, network analysis, link analysis, etc. For example, how important a person is within a social network, or, how important an author is according to the scientific literature network (e.g. the impact factor or the h-index, see Garfield (2005); Wendl (2007)), or, how prestigious a web page is in the word wide web (e.g. page rank or hits, see Brin and Page (1998); Kleinberg (1999)). These are the kind of questions graph analysis tools try to answer.

The most basic way to assign a centrality score to a node of an undirected graph is by using its degree: the number of transitions on a node. When the graph is directed we distinguish between the *in-degree* (i.e. incoming links) and the *out-degree* (i.e. outgoing links). As detailed by Wasserman and Faust (1994), the in-degree of a node measures the importance of this node in the case of a directed graph. In this case, one speaks about the *prestige* of an edge (i.e. a node in the graph). However a true *centrality measure* could be calculated on directed (e.g. the nodes out-degree) and undirected graphs as well (e.g. the nodes degree). Therefore, the famous page rank score — implemented by Google — is considered as a prestige measure of a web page since page rank is not meaningful when the network is undirected (i.e. it reduces to node degrees).

*In the remainder of this section, we investigate the case of node analysis but the same proof holds for link analysis. Nevertheless, the personal contribution material of this thesis will be developed for links analysis as well.

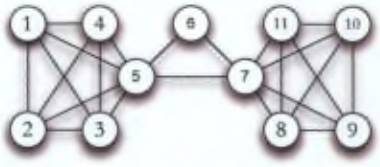


Figure 4.1: **An example network introduced by Newman (2005) .**

Usually, on a undirected graph, the centrality node degree is defined as

$$\text{cent}_D(k) = \sum_{k' \in S(k)} a_{kk'} \,, \tag{4.1}$$

where $S(k)$ is the set of successors of node k and $\mathbf{A}$ is the adjacency matrix associated to the graph. Looking at the network in the Figure 4.1, proposed by Newman (2005), the nodes 5 and 7 obtain a cent_D of 5, nodes $1, 2, 3, 4, 8, 9, 10, 11$ get a cent_D of 4 while the last node 6 achieves a cent_D of 2. While one can easily admit that nodes 5 and 7 are the more central nodes, one can question why nodes $1, 2, 9$ or 10 are more central than node 6 in the network. By visualizing the graph, it seems that node 6 is more central in term of spatial position than node $1, 2, 9$ or 10. Clearly, the degree is a local measure which is computed from the direct local neighborhood of a node. Hence, it can not capture spatial or topologic relations beyond the direct nearby nodes. In order to take into account topologic relations more spread into the network, Bavelas (1950) introduced a centrality measure based on the *closeness*. This measure quantifies how *close* a node is to all the others within the network. The central nodes are close when they can reach all the nodes in minimum steps. In other words, the *shortest-paths* linking the node of interest to the other nodes of the network must be as short as possible. If geodesics (i.e. shortest paths) increase in length, the closeness centrality of the node should decrease. Hence, geodesics (i.e. shortest paths) distances will have to be weighted inversely. The node closeness centrality on undirected graph is usually defined as

$$\text{cent}_C(k) = \frac{|E| - 1}{\sum_{k' \in E, k' \neq k} d(k, k')} \,, \tag{4.2}$$

where $d(k, k')$ is the shortest-path length for going from node k to node k' and $|E|$ is the number of nodes in the network. The closeness value ranges between 0 and 1. It equals unity when the node is adjacent to all the others (e.g. star topology). Note that closeness is undefined on unconnected components of a graph. According to this closeness, the more central nodes in the example of Figure 4.1 are still node 5 and 7 with a cent_C of 0.71, followed by node 6 with a cent_C of 0.55 and finally all the other nodes achieve a cent_C of 0.50. Assessing closeness, for all the nodes of a network, requires computing the geodesics distances between all pairs of nodes. This

can be done using the Floyd-Warshall's algorithm with a time complexity of $\mathcal{O}(|E|^3)$. However, since the adjacency matrix $\mathbf{A}$ is generally sparse, one will favorably make use of the Johnson's algorithm which time complexity is $\mathcal{O}(|V|^2 \log |V| + |V||E|)$.
Another famous centrality measure is known as the *betweennes*. The main idea behind betweenness is that a node is central in a network if it lies *between* the other nodes on their geodesics. Consequently, to have a large *betweenness* centrality, the node must be between many nodes via their geodesics. The Freeman's betweenness (see, e.g., Wasserman and Faust, 1994, Section 5.2.3) is defined for directed graphs as

$$\text{cent}_{\mathcal{B}}(k) = \frac{\sum_{i<j, i\neq k, j\neq k} g_{ij}(k)/g_{ij}}{(|E|-1)(|E|-2)/2}, \tag{4.3}$$

where g_{ij} is the number of geodesics linking node i to node j, $g_{ij}(k)$ is the number of geodesics linking these same two nodes but going through node k and $|E|$ is the number of edges in the graph. In the example graph of Figure 4.1 all the nodes except nodes 5 and 7 have a null betweenness, since these nodes never lay as intermediary nodes on any geodesics. Meanwhile, nodes 5 and 7 obtain a Freeman's betweenness score of 0.53. Indeed, these two nodes appear on geodesics going from the left-hand side to right-hand side of the graph (and conversely). Newman (2005) observes that by counting only shortest-paths, the Freeman's betweenness assumes that information spreads only along geodesics. Hence, he proposes a betweenness measure based on random walks that includes the contribution of all paths connecting nodes. The Newman's betweenneess is defined as the average of the current flow over all source-target pairs:

$$\text{cent}_{\mathcal{N}}(k) = \frac{\sum_{k, s<t} i_k^{(st)}}{|E|(|E|-1)/2} \tag{4.4}$$

$i_k^{(st)}$ represents the current flowing through the ith vertex, it is given by half of the sum of the absolute values of the current flowing along the edges incident to that vertex. When the ith vertex corresponds to the source (i.e. vertex s) or target (i.e. vertex t) vertex then the current flow is fixed to one unit (i.e. the total current injected and removed from the network). For details about how to compute the current $i_k^{(st)}$ please refer to the paper of Newman (2005). The current will flow along all paths from source to target, but more along shorter than longer ones, the shorter ones offering less resistance than the longer.
In the same example graph (Figure 4.1), Newman's betweenness obtains for nodes 5 and 7 a $\text{cent}_{\mathcal{N}}$ of 0.67, for node 6 a $\text{cent}_{\mathcal{N}}$ of 0.33 and the other nodes achieve a $\text{cent}_{\mathcal{N}}$ 0.269. While in Freeman's betweenness node 6 was directly disqualified, using Newman's betweenness this node ranked better than East and West nodes of the example network. Clearly, this example shows that taking into account alternative

paths can be worthwhile as well.

Contributions and organization of this Chapter

This chapter has two main contributions:

- It introduces a well-founded **covariance** as well as a **betweenness** measure between nodes of a weighted directed graph. The resulting covariance matrix defines a valid kernel on a graph.
- It shows how these covariance and betweenness measures can be computed efficiently from the immediate costs associated to each arc by inverting a $n \times n$ matrix, where n is the number of nodes in the graph.

These contributions has been published in the following journal paper:

Amin Mantrach, Luh Yen, Jerome Callut, Kevin Francoisse, Masashi Shimbo, and Marco Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1112Ð1126, 2010. ISSN 0162-8828

In the subsequent Section, we present state-of-the art kernels and advanced similarity measures on graphs. In Section 4.2, based on the randomized shortest-paths framework introduced in Chapter 3, we introduce two novels measures for directed — not necessarily strongly connected — graphs that take into account alternative paths as well. Section 4.3 produces a clear algorithm to compute these measures. This algorithm is then applied on small examples.

4.1 Similarity between Nodes

Similarity between nodes is also called relatedness in the literature and the most well-known quantities measuring relatedness are co-citation (see, e.g., Small, 1973) and bibliographic coupling (see, e.g., Kessler, 1963). In a corpus, co-citation coupling defines relatedness between documents as the number of other documents citing them both. Bibliographic coupling defines relatedness between two documents of a corpus as the number of common references cited by the two. On the other hand, the standard correlation measure between two nodes of a network (see, e.g., Wasserman and Faust, 1994) is the inner product between the normalized node vectors. Here, each node vector simply contains as elements the weights of the direct links of this node to the other nodes in the graph. This measure is therefore closely related to co-citation but unlike co-citation it takes into account the weights on the links.

4.1.1 Adjacency and Laplacian matrix of a graph

Following Fouss et al. (2007b), consider a weighted, undirected, graph $\mathcal{G}$ 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 $\mathbf{A}$ 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 $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D} = \mathbf{Diag}(a_{i\bullet})$ is the degree matrix, with diagonal entries $d_{ii} = [\mathbf{D}]_{ii} = a_{i\bullet} = \sum_{j=1}^{n} a_{ij}$, if the graph has n nodes in total. The volume of the graph is defined as $V_G = vol(G) = \sum_{i=1}^{n} d_{ii} = \sum_{i,j=1}^{n} a_{ij}$. 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 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 $\mathbf{L}$ has rank $n - 1$ (see, e.g., Chung, 1997). If $\mathbf{e}$ is a column vector all of whose elements are "1" (i.e., $\mathbf{e} = [1, 1, \ldots, 1]^{\mathsf{T}}$, where T denotes the matrix transpose) and $\mathbf{0}$ is a column vector all of whose elements are "0", $\mathbf{Le} = \mathbf{0}$ and $\mathbf{e}^{\mathsf{T}}\mathbf{L} = \mathbf{0}^{\mathsf{T}}$ hold: $\mathbf{L}$ is doubly centered. The null space of $\mathbf{L}$ is thus the one-dimensional space spanned by $\mathbf{e}$. Moreover, $\mathbf{L}$ is symmetric and positive semidefinite (see for instance Chung (1997)). Notice that, if the graph is not connected, the graph can be decomposed into closed subsets of nodes which are independent (there is no communication between them), each closed subset being irreducible, and the analysis can be applied independently on these closed subsets.

Let's consider the example network of Figure 4.1. The corresponding adjacency matrix $\mathbf{A}$ is

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Since the graph is undirected the adjacency matrix is symmetric. The diagonal matrix

$\mathbf{D}$ containing as diagonal the node degrees is

$$\mathbf{D} = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \end{bmatrix}$$

Then, the corresponding laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is

$$\mathbf{L} = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 4 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 6 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 6 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 4 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 4 \end{bmatrix}$$

4.1.2 The regularized Laplacian kernel

As stated by Fouss et al. (2007b), consider a weighted, undirected, graph $\mathcal{G}$. The **regularized Laplacian kernel** associated to the undirected graph (see, e.g., Chebotarev and Shamis, 1997, 1998; Ito et al., 2005; Smola and Kondor, 2003) is derived from the laplacian matrix $\mathbf{L}$ in the following way:

$$\mathbf{K}_{\mathrm{L}} = \sum_{k=0}^{\infty} \alpha^k (-\mathbf{L})^k = (\mathbf{I} + \alpha \mathbf{L})^{-1} \tag{4.5}$$

where $0 < \alpha < ||\mathbf{L}||_2^{-1}$ and $||\mathbf{L}||_2$ is the spectral radius of $\mathbf{L}$. $\mathbf{K}_{\mathrm{L}}$ is clearly positive definite. The regularized Laplacian kernel has been shown to provide competitive performance for a link prediction task (see, e.g., Ito et al., 2005).

This similarity measure has an interesting interpretation in terms of the matrix-forest

theorem (see, Chebotarev and Shamis, 1997, 1998). Let F^i be the set of all spanning forests rooted at node i of graph $\mathcal{G}$ 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 $\mathcal{G}$ that has the same nodes as $\mathcal{G}$ and one marked node (a root) in each component. It is shown in Chebotarev and Shamis (1997, 1998) that the matrix $(\mathbf{I}+\mathbf{L})^{-1}$ exists and that $[(\mathbf{I}+\mathbf{L})^{-1}]_{ij} = \epsilon(F^{ij})/\epsilon(F^i)$ 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 $(\mathbf{I}+\alpha\mathbf{L})^{-1}$ with a parameter α weighting the number of edges belonging to the forests as well as limiting the size of forests (in terms of number of edges (see Chebotarev and Shamis, 1997, 1998)).

Let's consider the example network of Figure 4.1. The spectral radius of the corresponding laplacian matrix is 0.760. α has to be chosen between 0 and $\frac{1}{0.760}$. For an α value of 0.1 we obtain the kernel of Figure 4.2:

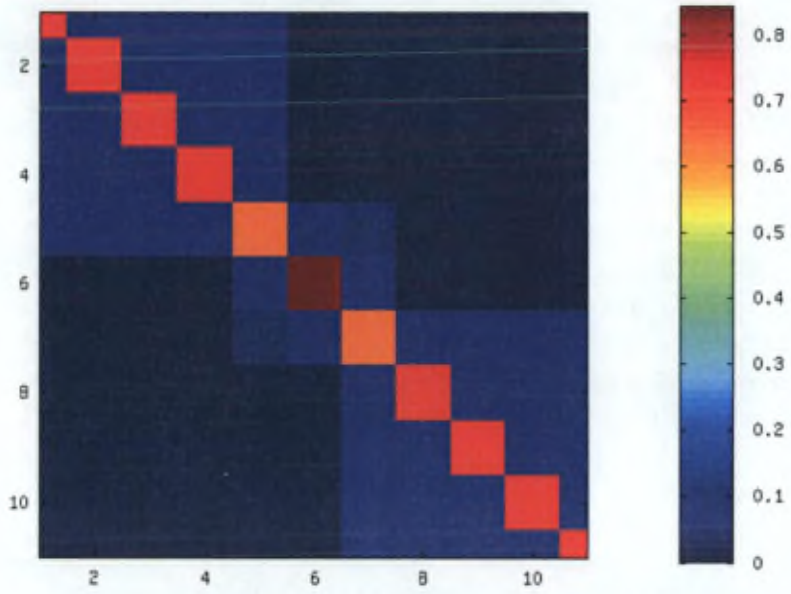


Figure 4.2: **Regularized Laplacian Kernel for** $\alpha = 0.1$ **.**

4.1.3 The commute-time kernel

As stated by Fouss et al. (2007b), the "commute-time" (CT) kernel has been introduced by Fouss et al. (2007a); Saerens et al. (2004) and was inspired by the already mentioned work of Klein and Randic (1993) and Chandra et al. (1989). It takes its name from the average commute time, which is defined as the average number of steps

a random walker, starting from a given node, will take before entering another node for the first time, and go back to the starting node. Hence, this measure is defined only for undirected and connected graphs. Indeed, if the graph is unconnected it will be impossible for a random walker to reach the destination node, if it is located in an other component of the graph. The CT kernel is defined as the inner product in a Euclidean space where the nodes are exactly separated by the commute-time distance.

We consider a Markov model for which the transition probabilities are provided by $p_{ij} = a_{ij}/a_{i\bullet}$ with $a_{i\bullet} = \sum_{j=1}^{n} a_{ij}$. In other words, to any state or node $X_t = i$, we associate a probability of jumping to an adjacent node $X_{t+1} = j$ that is proportional to the weight $a_{ij} \geqslant 0$ of the edge connecting i and j (this corresponds to a standard random-walk model on a graph).

The average commute time can be computed as proposed in Fouss et al. (2007a); Klein and Randic (1993); Saerens et al. (2004):

$$n(i,j) = V_G\,(\mathbf{e}_i - \mathbf{e}_j)^{\mathsf{T}}\mathbf{L}^{+}(\mathbf{e}_i - \mathbf{e}_j) \tag{4.6}$$

where every node i of the graph is represented by a basis vector $\mathbf{e}_i$ in the Euclidean space $\Re^n$ and V_G is the volume of the graph. $\mathbf{L}^+$ is the Moore-Penrose pseudoinverse of the Laplacian matrix of the graph and it is positive semidefinite. Thus, Equation (4.6) is a Mahalanobis distance between the nodes of the graph and is referred to as the "commute-time distance" or the "resistance distance" because of a close analogy with the effective resistance in electrical networks (see, e.g., Chandra et al., 1989; Fouss et al., 2007a; Hirai et al., 2005; Klein and Randic, 1993).

It can be shown that the elements of $\mathbf{L}^+$ are inner products of the node vectors in the Euclidean space where these node vectors are exactly separated by commute-time distances. In other words, the elements of $\mathbf{L}^+$ can be viewed as similarity measures between nodes. Hence the **commute-time kernel** $\mathbf{K}_{\mathrm{CT}}$ is defined as

$$\mathbf{K}_{\mathrm{CT}} = \mathbf{L}^{+} \tag{4.7}$$

with no parameter tuning necessary. There is of course a close relationship between the commute-time kernel and the regularized Laplacian kernel. Indeed, if $\mathbf{L}$ has eigenvalues λ_i (in decreasing order), $(\mathbf{L} + \alpha^{-1}\mathbf{I})$ has corresponding eigenvalues $(\lambda_i + \alpha^{-1})$ and both matrices share the same eigenvectors $\mathbf{u}_i$. Remember that $\mathbf{L}$ and its Moore-Penrose pseudoinverse $\mathbf{L}^+$ have the same set of eigenvectors and inverse eigenvalues (zero eigenvalues are equal to zero in both cases). Moreover, the Laplacian matrix of a connected graph has rank $n-1$ and therefore has exactly one eigenvalue equal to 0 (see, e.g., Chung, 1997). Thus a spectral decomposition of the corresponding kernel yields $\mathbf{L}^+ = \sum_{i=1}^{n-1} \lambda_i^{-1}\mathbf{u}_i\mathbf{u}_i^{\mathsf{T}}$ (by definition of the Moore-Penrose pseudoinverse) and $(\mathbf{I} + \alpha\mathbf{L})^{-1} = \alpha^{-1}\sum_{i=1}^{n}(\lambda_i + \alpha^{-1})^{-1}\mathbf{u}_i\mathbf{u}_i^{\mathsf{T}}$. Since the normalized eigenvector of $\mathbf{L}$ corresponding to eigenvalue $\lambda_n = 0$ is $\mathbf{u}_n = \mathbf{e}/\sqrt{n}$, the nth term for the commute-time

kernel is simply 0, while it is $\mathbf{e}\mathbf{e}^T/n$ 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 α is large, $\lambda_i + \alpha^{-1}$ will be close to λ_i, so that the two kernels essentially differ by a multiplication factor and a constant value added to all the elements of the matrix.

An interesting method allowing to efficiently compute truncated commute-time neighbors appears in (see, e.g., Sarkar and Moore, 2007). Almost at the same period, Qiu and Hancock (2005, 2007), Ham et al. (2004), Yen et al. (2005) as well as Brand (2005) defined the same CT embedding, preserving the commute-time distance, and applied it to image segmentation and multi-body motion tracking (see, e.g., Qiu and Hancock, 2005, 2007), to dimensionality reduction of manifolds (see, e.g., Ham et al., 2004), to clustering (see, e.g., Yen et al., 2005) as well as to collaborative filtering (see, e.g., Brand, 2005; Fouss et al., 2007a), with interesting results. The commute-time kernel is also closely related to the "Fiedler vector" (see, e.g., Fiedler, 1975; Mohar, 1992), widely used for graph partitioning (see, e.g., Chan et al., 1997; Pothen et al., 1990) or clustering (see, e.g., Donetti and Munoz, 2004; Yen et al., 2007, 2009), as detailed in Fouss et al. (2007a). An electrical interpretation of the elements of the CT kernel is provided in Yen et al. (2009). Families of dissimilarity measures subsuming both the shortest-path distance and the commute-time distance were recently proposed by Yen et al. (2008) and by Chebotarev (2008). Notice that the dissimilarity defined by Chebotarev (2008) is a distance; that is, it verifies all the properties of a distance, including the triangular inequality (which is not the case for Yen et al. (2008)).

On the other hand, Zhou (2003b,a) uses the average first passage time between two nodes as a dissimilarity index in order to cluster them. He studies various greedy clustering techniques based on this dissimilarity index. Another similarity measure related to the average fist-passage time appears in Tong et al. (2007). It is defined as the escape probability, that is the probability that a random walker starting from one node will visit the other node, before returning to the starting node. The resulting similarity is directed and closely related to the effective conductance between the two nodes. Also related is the measure investigated by Koren et al. (2006, 2007) where the authors propose to replace the effective conductance by a cycle-free effective conductance.

The commute-time kernel has been computed on the example network of Figure 4.1 and is shown in Figure 4.3.

4.1.4 The random-walk-with-restart similarity matrix

As stated by Fouss et al. (2007b), Pan et al. (2006) (see also Tong et al., 2006, 2008) recently introduced a similarity matrix between nodes inspired by the well-known PageRank algorithm (see Brin and Page, 1998; Page et al., 1999). This model has been applied to various interesting applications, including center-piece subgraph discovery and content-based image retrieval (see, e.g., Pan et al., 2006; Tong et al., 2006,

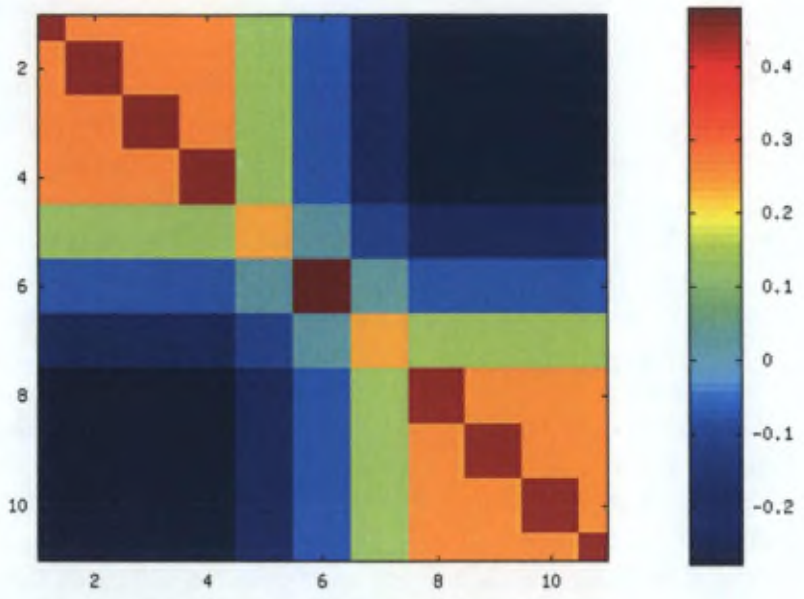


Figure 4.3: **Commute-time Kernel matrix.**

2008). The same idea was also introduced by Gori and Pucci (2006b,a) in the context of collaborative recommendation.

Like the diffusion kernel, the model considers a random walker jumping from some node i and to some neighbor node j with a probability proportional (apart from normalization) to the edge weight $p_{ij} = \mathrm{P}(s(t+1) = j | s(t) = i) = a_{ij}/a_{i\bullet}$. In addition, at each step of the random walk, the random walker has some probability $(1-\alpha)$ to return 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

$$\begin{cases} \mathbf{x}(0) = \mathbf{e}_i \\ \mathbf{x}(t+1) = \alpha \mathbf{P}^{\mathsf{T}} \mathbf{x}(t) + (1-\alpha)\mathbf{e}_i \end{cases} \tag{4.8}$$

Considering the steady-state solution $\mathbf{x}(t+1) = \mathbf{x}(t) = \mathbf{x}$ and extracting the probability distribution $\mathbf{x}$ of finding the random walker on each node when starting from node i yields

$$\mathbf{x} = (1-\alpha)\left(\mathbf{I} - \alpha \mathbf{P}^{\mathsf{T}}\right)^{-1} \mathbf{e}_i \tag{4.9}$$

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

Now, since $\mathbf{x}$ is column i of matrix $\left(\mathbf{I} - \alpha \mathbf{P}^{\mathsf{T}}\right)^{-1}$, the **random-walk-with-restart**

($\mathbf{K}_{\text{RWR}}$) **matrix** (see, e.g., Gori and Pucci, 2006b,a; Pan et al., 2006; Tong et al., 2006, 2008) will be defined as the matrix whose i'th row contains the similarities to node i, as usual for similarity matrices.

Since matrix $(\mathbf{I}-\alpha\mathbf{P}^{\mathrm{T}})^{-1}$ is not symmetric, we transpose it to get the correct similarity matrix:

$$\mathbf{K}_{\text{RWR}} = (\mathbf{I} - \alpha\mathbf{P})^{-1} \tag{4.10}$$

$\mathbf{K}_{\text{RWR}}$ can be rewritten as

$$\mathbf{K}_{\text{RWR}} = (\mathbf{D}^{-1}(\mathbf{D} - \alpha\mathbf{A}))^{-1} = (\mathbf{D} - \alpha\mathbf{A})^{-1}\mathbf{D} \tag{4.11}$$

since $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$. Notice that this measure in not symmetric. Indeed, having a network the probability starting from node i to reach node j is generally different than the probability starting from node j to reach node i. For example, Figure 4.4 shows the random walk with restart kernel computed on the Newman's example network (Figure 4.1). When starting from any node, the similarity to node 6 is higher than the probability to reach any of these nodes when starting from node 6 itself. Indeed, this is due to the central position of node 6 in the network.

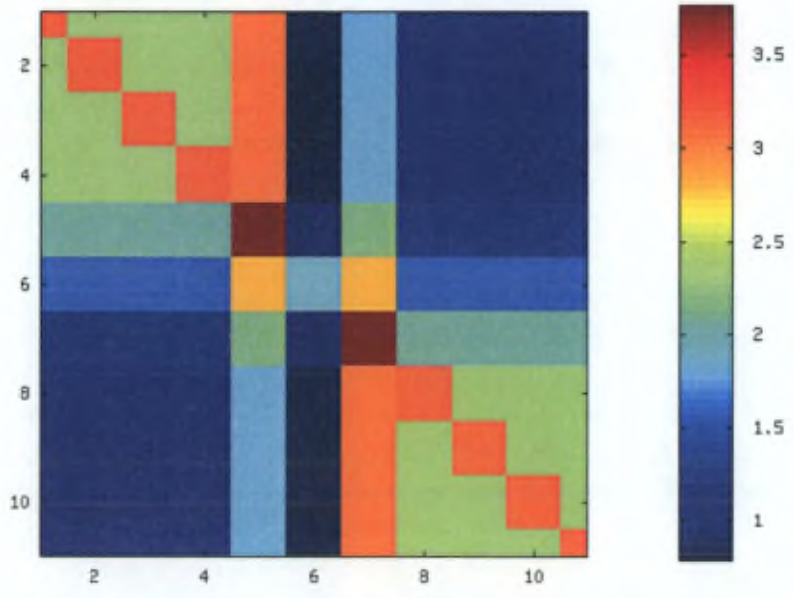


Figure 4.4: **Random walk with restart similarity matrix for** $\alpha = 0.95$ **.**

4.1.5 Other similarity measures

Other sophisticated measures have been proposed as well. For instance, Klein and Randic (1993) proposed to use the effective resistance between two nodes as a meaningful distance measure. They call this quantity the resistance distance between nodes. Indeed, it can be shown that the effective resistance is a Euclidean distance (see, e.g., Bapat, 1999; Gobel and Jagers, 1974; Klein and Randic, 1993; Lovasz, 1996). The close link between the effective resistance and the commute time of a random walker on the graph was established by Chandra et al. (1989) while the links between the Laplacian matrix and the commute-time (as well as the Fiedler vector) were studied by Saerens et al. (2004). Therefore, the resistance distance is sometimes called the commute-time distance.

The exponential and von Neumann diffusion kernels, based this time on the adjacency matrix, are introduced in Kandola et al. (2002); Shawe-Taylor and Cristianini (2004). The defined kernel matrices are computed through a power series of the adjacency matrix of the graph; they are closely related to graph regularization models (see, e.g., Kondor and Lafferty, 2002).

Moreover, some authors recently considered similarity measures based on random-walks or electrical concepts (for a nice introduction to this topic, see, e.g., Doyle and Snell, 1984). For instance, Harel and Koren (2001) investigated the possibility of clustering data according to some random-walk related quantities, such as the probability of visiting a node before returning to the starting node. They showed that their algorithm is able to cluster arbitrary nonconvex shapes. White and Smyth (2003) investigated the use of the average first-passage time as a similarity measure between nodes. Their purpose was to generalize the random-walk approach of Brin and Page (1998); Page et al. (1999) by capturing a concept of "relative centrality" of a given node with respect to some other node of interest. A recent study, comparing several measures for analyzing the proximity of nodes in a graph in the framework of co-authorship networks, is presented by Liben-Nowell and Kleinberg (2007).

On the other hand, Kondor and Lafferty (2002) as well as Smola and Kondor (2003) defined a graph regularization model related to the graph principal components analysis introduced by Saerens et al. (2004). This model results in the definition of a family of kernels on a graph that provide similarities between nodes, just as any other graph kernel (see, e.g., Shawe-Taylor and Cristianini, 2004). An interesting attempt to learn the regularization operator in the context of semi-supervised learning can be found the book chapter of (Zhu et al., 2006). The result is a kernel on a graph maximizing kernel alignment to the labeled data, in a semi-supervised setting. Another approach has been investigated by Palmer and Faloutsos (see, e.g., 2003) who define a similarity function between categorical attributes, called "refined escape probability", based on random walks and electrical networks. They show that this quantity provides a reasonably

good measure for clustering and classifying categorical attributes.

In two recent papers Nadler et al. (2005, 2006), as well as Pons and Latapy (2005, 2006) proposed a well-formulated distance measure between nodes of a graph based on a diffusion process, called the "diffusion distance". A valid kernel, called the "Markov diffusion kernel" has been derived from this diffusion distance by Fouss et al. (2006). An application of the diffusion distance to dimensionality reduction and graph visualization appears in Lafon and Lee (2006). The natural embedding induced by the diffusion distance is called the "diffusion map" by Nadler et al. (2005, 2006). Moreover, Pons and Latapy (2005, 2006) defined a hierarchical clustering approach for clustering the nodes according to the squared diffusion distance.

Finally, a similarity between nodes based on the number of different paths connecting two nodes, and therefore on the maximum flow/minimum cut, is studied by Lu et al. (2006). This measure has been tested in two collaborative recommendation tasks by Fouss et al. (2007a), but did not perform well in this context. Finally, Tahbaz and Jadbabaie (2006) introduce a one-parameter family of algorithms that recovers both the Bellman-Ford procedure for finding shortest paths as well as the iterative algorithm for computing the average fist-passage time. It is based on heuristic grounds and not on a well-defined cost function to optimize.

There are also several attempts to generalize graph kernels to directed graphs. For instance, an extension of the Laplacian matrix to directed graphs is proposed by Chung (2005) while an extension of the regularized Laplacian kernel to directed graphs has been proposed by Agaev and Chebotarev (2000, 2001). Zhou et al. (2005) used the regularized normalized Laplacian matrix defined by Chung (2005) in the context of semi-supervised classification of labeled nodes of a directed graph while Chen et al. (2007) used the same kernel matrix, but this time unnormalized, for directed graph embedding. Zhao et al. (2007) propose a directed contextual distance and define a directed graph from which the Laplacian matrix is computed. It is then used for ranking and clustering images.

Most of the mentioned approaches (except those described in the just previous paragraph) are restricted to the analysis of undirected graphs. For instance, the Laplacian-based techniques assume, most of the time, a symmetric adjacency matrix, that is, an undirected graph. Many approaches further assume a connected graph having one single connected component. Finally, some approaches consider only aperiodic Markov chains (such as the diffusion map method), do not allow absorbing nodes or lack a clear, intuitive, interpretation. The SoP approach developed further in this section does not suffer from any of these problems while remaining competitive in terms of performance, at least for the investigated semi-supervised classification tasks (see the experimental section of next Chapter, Section 5.3). However, the main drawback of the SoP approach, as any similarity measure between nodes of a graph, is that it does not scale well on large graphs when the entire kernel matrix is needed. This issue will be

investigated in Chapter 6. Still another weakness of the SoP approach is the fact that it depends on a hyper-parameter that has to be tuned. However, in contrast with some of the presented approaches, the SoP hyper-parameter has a clear interpretation.

4.2 The Sum-over-Paths Betweenness and Covariance Measures

Let's now introduce two novel measures that are contributions of this thesis. Indeed, following Saerens et al.'s arguments (2009) — introduced in Section 3.2 — it is now shown that a novel betweenness centrality and covariance relatedness can be computed from a quantity, appearing in the denominator of Equation (3.6) defined as

$$\mathcal{Z} = \sum_{\wp \in \mathcal{R}} \exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right], \tag{4.12}$$

and which corresponds to the **partition function** in statistical physics (see Jaynes (1957) or any textbook in statistical physics; for instance Reichl (1998); Schrodinger (1952)). Let us further define the **free energy** F as

$$F = -\frac{1}{\theta}\ln(\mathcal{Z}) = -T\ln(\mathcal{Z}) \tag{4.13}$$

where $T = 1/\theta$ is the **temperature** of the system.

Indeed, let us first show how the **expected energy** (or **expected cost**) can be computed from the partition function:

$$\begin{aligned}
\overline{E} &= \frac{\partial(-\ln(\mathcal{Z}))}{\partial\theta} && (4.14)\\
&= \sum_{\wp \in \mathcal{R}} \frac{\exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]}{\mathcal{Z}} E(\wp) && (4.15)\\
&= \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, E(\wp) && (4.16)
\end{aligned}$$

The **expected number of transitions** through the link $k \to k'$ can also be easily computed:

$$\begin{aligned}
\overline{\eta}(k,k') &= \frac{\partial F}{\partial c_{kk'}} = \frac{1}{\theta}\frac{\partial(-\ln \mathcal{Z})}{\partial c_{kk'}} && (4.17)\\
&= \sum_{\wp \in \mathcal{R}} \frac{\exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]}{\mathcal{Z}} \delta(\wp; k, k') && (4.18)\\
&= \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, \delta(\wp; k, k') && (4.19)
\end{aligned}$$

where $\delta(\wp; k, k')$ indicates the number of times the link $k \to k'$ is present in path $\wp$ and thus the number of times the link is traversed within this path.

On the other hand, the **expected number of passages** in node k, which defines the **betweenness measure**, is

$$\text{bet}(k) = \overline{n}_k = \sum_{l=1}^{n} \overline{\eta}(l, k) \tag{4.20}$$

and corresponds to the sum of incoming transitions in node k. This is the first quantity of interest.

Furthermore, the expected number of times the link $k \to k'$ and the link $l \to l'$ are traversed together along a path is

$$\overline{\eta}(k, k'; l, l') = \frac{1}{\theta^2} \frac{\partial^2 (\ln \mathcal{Z})}{\partial c_{ll'} \partial c_{kk'}} \tag{4.21}$$

$$= \sum_{\wp \in \mathcal{R}} \frac{\exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]}{\mathcal{Z}} \delta(\wp; k, k') \delta(\wp; l, l') - \left[\sum_{\wp \in \mathcal{R}} \frac{\exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]}{\mathcal{Z}} \delta(\wp; k, k') \right] \times \left[\sum_{\wp \in \mathcal{R}} \frac{\exp\left[-\theta E(\wp) + \ln \mathrm{P}^{\mathrm{ref}}(\wp)\right]}{\mathcal{Z}} \delta(\wp; l, l') \right] \tag{4.22}$$

$$= \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, \delta(\wp; k, k') \delta(\wp; l, l') - \left[\sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, \delta(\wp; k, k') \right] \left[\sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, \delta(\wp; l, l') \right] \tag{4.23}$$

$$= \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, \delta(\wp; k, k') \delta(\wp; l, l') - \overline{\eta}(k, k')\, \overline{\eta}(l, l') \tag{4.24}$$

and this quantity is a measure of covariance between link $k \to k'$ and link $l \to l'$.

Finally, the **covariance measure** between node k' and node l' is simply defined from Equation (4.24) as

$$\text{cov}(k', l') = \sum_{k,l=1}^{n} \overline{\eta}(k, k'; l, l') \tag{4.25}$$

$$= \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, \delta(\wp; k') \delta(\wp; l') - \overline{n}_{k'}\, \overline{n}_{l'} \tag{4.26}$$

$$= \sum_{\wp \in \mathcal{R}} \mathrm{P}(\wp)\, (\delta(\wp; k') - \overline{n}_{k'})(\delta(\wp; l') - \overline{n}_{l'}) \tag{4.27}$$

which is the second quantity of interest. Here, $\delta(\wp; k') = \sum_k \delta(\wp; k, k')$ indicates the

number of times node k' is visited on path $\wp$. Obviously, Equation (4.27) defines a valid kernel on a graph since $\mathrm{cov}(k', l')$ is an inner product in the paths space.
Of course, one single parameter θ could not be adequate for all regions of a large graph. In a certain sense, the θ parameter regulates the subset of paths for which the probability mass is significant, in function of the total cost of the path (see Figure 4.6 for an illustration of this property). For a dense, highly connected community, large values of θ should be sufficient (only short paths are considered).
Note that if there are different connected components, the covariance between the nodes of the first connected component and the second one will be negative. This can be seen from Equation (4.26): if there is no path connecting node k' from a first connected component to node l' from a second connected component, the first term of Equation (4.26) cancels.

4.3 Computation of the Betweenness and Covariance Measures

Now that we have seen how to compute the partition function $\mathcal{Z}$, we will turn to the computation of the betweenness and the covariance measures that can be deduced from $\mathcal{Z}$ thanks to Equations (4.17), (4.20) , (4.21) and (4.25).
However, let us first derive two formulas that will be useful in the sequel. They involve the ith row (viewed as a column vector) and the jth column of $\mathbf{Z}$: $\mathbf{Z}^{\mathrm{T}}\mathbf{e}_i = \mathbf{z}_i^r$ and $\mathbf{Z}\mathbf{e}_j = \mathbf{z}_j^c$. In other words, the column vector $\mathbf{z}_i^r = (\mathbf{row}_i(\mathbf{Z}))^{\mathrm{T}}$ contains the elements of the ith row of matrix $\mathbf{Z}$ while the column vector $\mathbf{z}_j^c = \mathbf{col}_j(\mathbf{Z})$ contains the elements of the jth column of $\mathbf{Z}$.
These two quantities can easily be found by solving the linear systems of equations

$$(\mathbf{I} - \mathbf{W})^{\mathrm{T}}\mathbf{z}_i^r = \mathbf{e}_i \text{ and } (\mathbf{I} - \mathbf{W})\mathbf{z}_j^c = \mathbf{e}_j \tag{4.28}$$

Elementwise, these last equations (4.28) yield

$$z_{ik'} = \delta_{ik'} + \sum_{k \in P(k')} z_{ik} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\mathrm{ref}}\right] \tag{4.29}$$

and

$$z_{kj} = \delta_{kj} + \sum_{k' \in S(k)} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\mathrm{ref}}\right] z_{k'j} \tag{4.30}$$

where $P(k')$ is the set of predecessors of node k' and $S(k)$ is the set of successors of node k. When summing these last equations over i and j, we obtain

$$z_{\bullet k'} = \sum_{i=1}^{n} z_{ik'} = 1 + \sum_{k \in P(k')} z_{\bullet k} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\mathrm{ref}}\right] \tag{4.31}$$

$$z_{k\bullet} = \sum_{j=1}^{n} z_{kj} = 1 + \sum_{k' \in S(k)} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\text{ref}}\right] z_{k'\bullet} \tag{4.32}$$

These equations will be useful later for deriving the expression of the covariance measure (see the Appendix A).

We thus have to compute the derivatives of $\mathcal{Z}$ (Equation (3.13)) in terms of $c_{kk'}$ (see Equations (4.17) and (4.21)) in order to obtain the different quantities of interest, which is done in Appendix A. For the **expected number of passages** through the link $k \to k'$, we obtain

$$\begin{aligned} \overline{\eta}(k,k') &= \frac{\partial F}{\partial c_{kk'}} && (4.33)\\ &= \frac{\sum_{i,j=1}^{n} z_{ik} z_{k'j} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\text{ref}}\right]}{\mathcal{Z}} && (4.34)\\ &= \frac{z_{\bullet k} z_{k'\bullet} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\text{ref}}\right]}{\mathcal{Z}} && (4.35) \end{aligned}$$

Therefore, the **expected number of passages** through node k' (the **betweenness** of node k') is

$$\text{bet}(k') = \bar{n}_{k'} = \sum_{k=1}^{n} \overline{\eta}(k,k') = \frac{(z_{\bullet k'} - 1) z_{k'\bullet}}{\mathcal{Z}} \tag{4.36}$$

where we used Equation (4.31). The column vector containing the elements $\text{bet}(k')$ will be called $\mathbf{b}$.

Moreover, the **covariance** between node k and node l is (see Appendix A)

$$\begin{aligned} \text{cov}(k,l) = \frac{1}{\mathcal{Z}} \Big\{ & (z_{\bullet k} - 1) z_{k\bullet} \delta_{kl} + z_{k\bullet} (z_{\bullet l} - 1)(z_{lk} - \delta_{lk}) \\ & + z_{l\bullet} (z_{\bullet k} - 1)(z_{kl} - \delta_{kl}) \\ & - \frac{z_{k\bullet} z_{l\bullet} (z_{\bullet k} - 1)(z_{\bullet l} - 1)}{\mathcal{Z}} \Big\} \end{aligned} \tag{4.37}$$

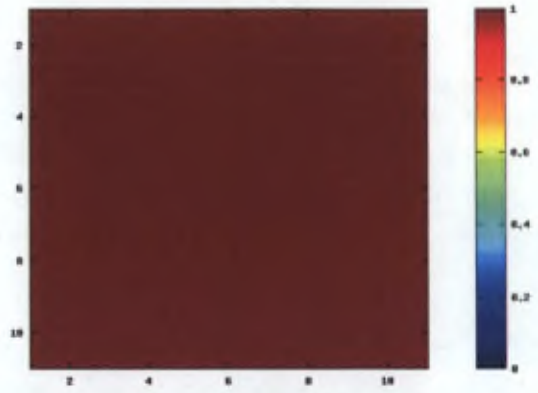
The matrix containing the elements $\text{cov}(k,l)$ will be denoted $\boldsymbol{\Sigma}$. On the other hand, the **correlation** between nodes k and l is

$$\text{cor}(k,l) = \frac{\text{cov}(k,l)}{\sqrt{\text{cov}(k,k)\text{cov}(l,l)}} \tag{4.38}$$

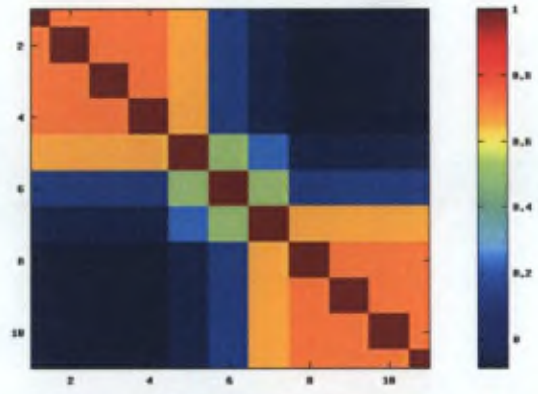
In Algorithm2[†], we present the corresponding algorithm for computing the betweenness as well as the covariance measures for a graph $\mathcal{G}$.

Notice that the covariance between one node of interest k and the remaining nodes (one column of the covariance matrix) can be obtained by solving four linear systems of equations instead: Equations (4.28) with $i = j = k$ (or, elementwise, Equations (4.29-4.30)), and Equations $(\mathbf{I} - \mathbf{W})\mathbf{z}^c = \mathbf{e}$, $(\mathbf{I} - \mathbf{W})^{\mathsf{T}}\mathbf{z}^r = \mathbf{e}$ (or, elementwise, Equations

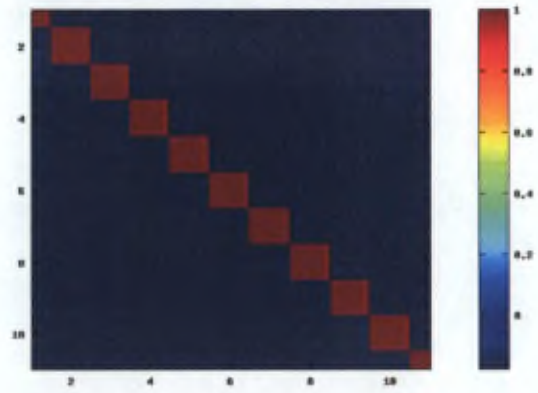
[†]The Matlab/Octave implementation of the algorithm can be downloaded from http://iridia.ulb.ac.be/~amantrac/pub/SoP_TPAMI.zip



(a)



(b)



(c)

Figure 4.5: **SoP correlation matrix computed on the graph of Figure 4.1 for respectively $\theta \to 0$ (a), $\theta = 0.1$ (b), and $\theta \to \infty$ (c). Dark red colors indicate high correlations while dark blue colors indicate low correlations.**

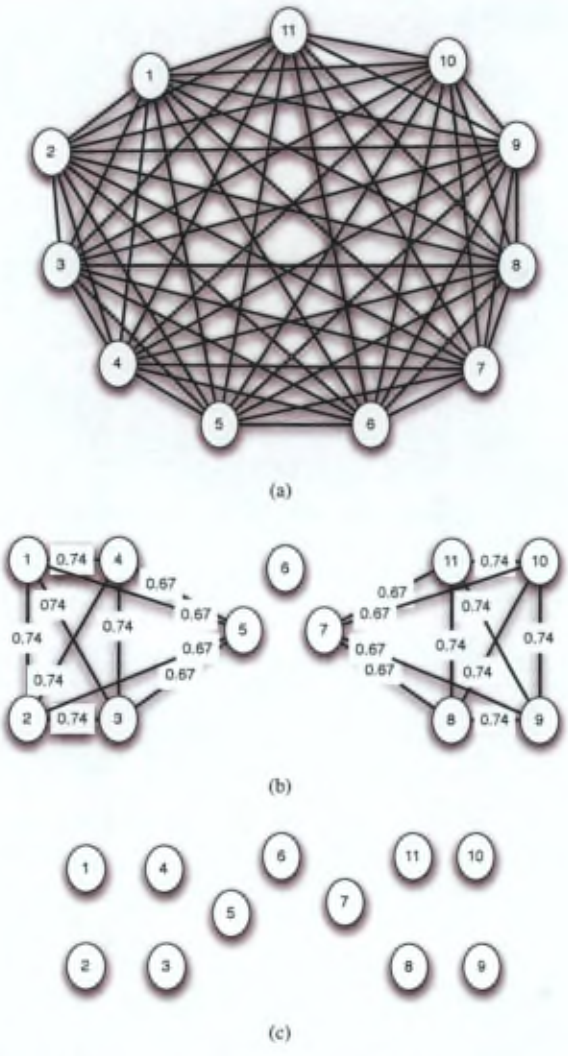


Figure 4.6: **Graph inferred from the SoP correlation matrix computed on the graph of Figure 4.1 for respectively $\theta \to 0$ (a), $\theta = 0.1$ (b), and $\theta \to \infty$ (c). Self-loops have been deliberately deleted. Weights on the graph (b) correspond to the SoP correlation.**

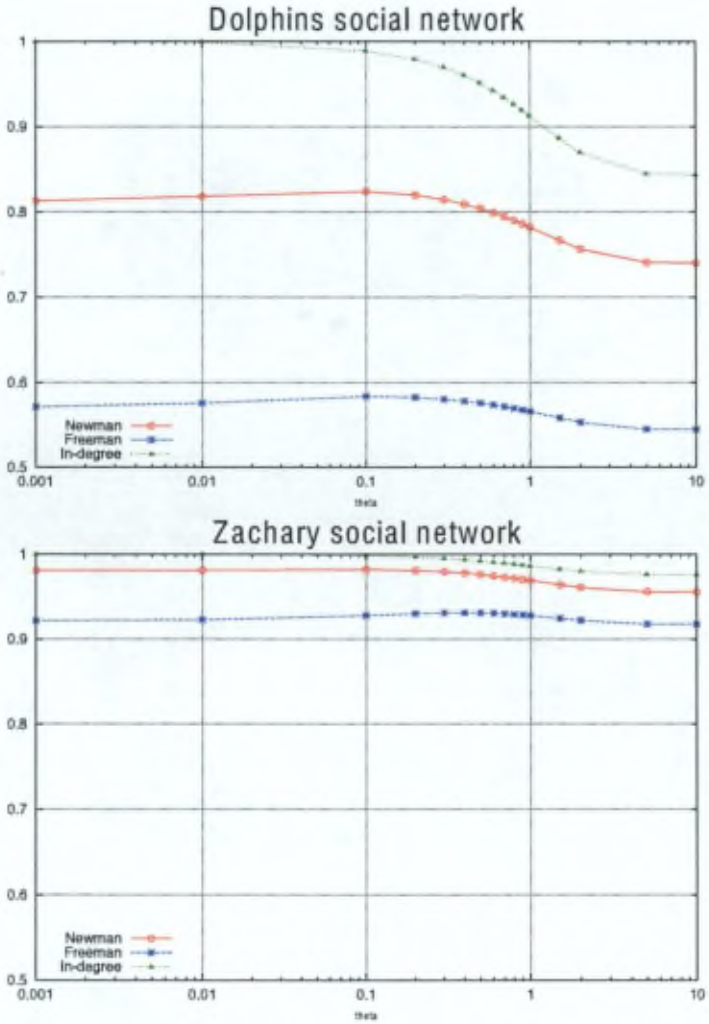


Figure 4.7: **Correlation of the SoP betweenness with the Newman's betwenness and the Freeman's betwenness for an increasing θ temperature .**

Algorithm 1 Computation of the betweenness vector and the covariance matrix between nodes.

Input:

- A graph $\mathcal{G}$ containing n nodes.
- $\theta > 0$: the parameter controlling the degree of randomness.
- $\mathbf{C}$: the $n \times n$ cost matrix associated to $\mathcal{G}$, containing elements $c_{kk'} > 0$.
- $\mathbf{P}^{\text{ref}}$: the $n \times n$ reference transition probabilities matrix.

Output:

- The **betweenness** vector $\mathbf{b}$ containing the betweenness of each node, $\text{bet}(k)$.
- The **covariance** matrix $\boldsymbol{\Sigma}$ between every pair of nodes, containing the elements, $\text{cov}(k,l)$.

1. $\mathbf{W} = \mathbf{P}^{\text{ref}} \circ \exp[-\theta \mathbf{C}]$, where $\circ$ is the elementwise product, and the exponential is taken elementwise
2. $\mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1}$ containing elements $z_{kk'}$
3. $z_{\bullet k'} = \sum_{k=1}^{n} z_{kk'}$, $z_{k\bullet} = \sum_{k'=1}^{n} z_{kk'}$, $z_{\bullet\bullet} = \sum_{k,k'=1}^{n} z_{kk'}$ for all k, k'
   $\mathcal{Z} = z_{\bullet\bullet} - n$
4. **for** $k = 1$ to n **do**
5. $\quad \text{bet}(k) = \dfrac{(z_{\bullet k} - 1) z_{k\bullet}}{\mathcal{Z}}$
6. $\quad$ **for** $l = k$ to n **do**
7. $\quad\quad \text{cov}(k,l) = \frac{1}{\mathcal{Z}} \Big\{ (z_{\bullet k} - 1) z_{k\bullet} \delta_{kl} + z_{k\bullet}(z_{\bullet l} - 1)(z_{lk} - \delta_{lk}) + z_{l\bullet}(z_{\bullet k} - 1)(z_{kl} - \delta_{kl}) - \dfrac{z_{k\bullet} z_{l\bullet}(z_{\bullet k} - 1)(z_{\bullet l} - 1)}{\mathcal{Z}} \Big\}$
8. $\quad\quad \text{cov}(l,k) = \text{cov}(k,l)$
9. $\quad$ **end for**
10. **end for**
11. **return** $\mathbf{b} = [\text{bet}(k)]$, $\boldsymbol{\Sigma} = [\text{cov}(k,l)]$

(4.31-4.32)) where $\mathbf{z}^c$ and $\mathbf{z}^r$ contain respectively the sum over columns (the row sums) and the sum over rows (the column sums) of $\mathbf{Z}$. $\mathbf{e}$ is a column vector full of 1's. Thus, the column vectors $\mathbf{z}^c$ and $\mathbf{z}^r$ respectively contain the elements $z_{k\bullet}$ and $z_{\bullet k'}$. All these linear systems of equations can be solved efficiently, especially when the matrix $\mathbf{W}$ is sparse (see, e.g. Davis, 2006). In particular, they could be solved iteratively. For instance, for Equations (4.28), we could iterate $\hat{\mathbf{z}}_i^r \leftarrow \mathbf{W}^{\mathsf{T}} \hat{\mathbf{z}}_i^r + \mathbf{e}_i$ and $\hat{\mathbf{z}}_j^c \leftarrow \mathbf{W} \hat{\mathbf{z}}_j^c + \mathbf{e}_j$ since we have $\rho(\mathbf{W}) < 1$ (see for instance Golub and Loan (1996); Meyer (2001)).

If, instead, all the elements of the covariance matrix have to be computed, the inversion of the matrix $(\mathbf{I} - \mathbf{W})$ is needed, which can be an issue when dealing with large graphs. First, the computation time for inverting the matrix is significant and, second, even if the original matrix is sparse, the inverse is usually dense and could not fit into main memory. One potential solution to this problem is to perform first an incomplete matrix factorization of the matrix. Indeed, if the graph is undirected, the matrix $\mathbf{D}(\mathbf{I} - \mathbf{W})$, where $\mathbf{D}$ is a diagonal matrix containing the row sums of the adjacency matrix related to the natural random walk, is positive definite. In this case, a low-rank (or incomplete)

Cholesky factorization can be computed efficiently Fine and Scheinberg (2001) and remains sparse if the original matrix is sparse. Once this factorization is computed, each column of the matrix inverse can be obtained by simple back-substitution – this technique therefore exploits the sparseness of the network. The design of algorithms able to mine large graphs by, for instance, limiting the length of the walks (such as, e.g., in Callut et al. (2008)), will be investigated in chapter 6.

Figure 4.6 shows the SoP correlation matrix obtained for different value of the entropy θ. The way a node is correlated to the others depends on the θ parameter. A high θ value means that we consider only shortest paths. Considering only shortest paths gives small chance to two nodes to appear together on the same path, which leads to a constantly low correlation. This can be observed on Figure 4.5(c) where we display the SoP matrix obtained for the graph of Figure 4.1. If we try to infer a graph, by only keeping links with a high correlation (e.g. > 0.6), no links remain in the obtained graph (see Figure 4.6(c)). Inversely, a θ close to 0 do not behave better for discriminating between nodes. The behavior can be observed on Figure 4.5(a) where all the nodes are highly correlated. This lead to the fully connected network of Figure 4.6(a).

To cluster better, we take a θ of 0.1, hence the left hand-side nodes are less correlated to right-hand side nodes. This leads to the 3 subgraphs of Figure 4.6(b), each subgraph corresponding to a cluster.

The correlation of the SoP betweenness with the Newman's betweenness and the Freeman's betweenness is reported on Figure 4.7. The Newman's betweenness is clearly more correlated to the SoP betweenness for θ values $\in [0; 0.1]$, giving importance to all paths as suggested by Newman. While for θ values greater than 0.1 the correlation with Newman's becomes lower since the SoP betweenness is biased through shortest-paths. On the other hand, the correlation with Freeman's stays the same for any value of θ. One would think that the SoP betweenness has to be more correlated to Freeman's betweenness for high values of θ since the measure favors shortest-paths. However, the SoP betweenness is different from the general definition of the betweenness defined only on paths of minimum length 2. Indeed, a node has to lie on paths of length 2 at least in order to be located between two other nodes. However, according to our definition, the first order derivate of the partition function consider all paths — of length 1 also. Hence, the SoP betweenness measures the expected number of passages in node k, within paths of any length (see Equation (4.19)). Another difference, lies in the fact that the SoP betweenness considers paths starting and ending in the same node as well. However, common betweenness is usually defined only for paths starting and ending on two different nodes of the network. Moreover, in the standard betweenness a node located at the beginning or at the end of a path is not counted, indeed a "between" node is not located at the bounds of a path, while in the SoP betweenness well. The Figure 4.7 report also the correlation of the SoP betweenness with the in-degrees of the nodes. For low θ values (< 0.1), by giving an equal importance to all paths, the SoP

betweenneess is fully correlated to the node's in-degree for graphs where the cost of each incoming transition into a node is identical.

Chapter 5

Within-Network Classification

Within-Network classification ranges among semi-supervised learning paradigm. The goal of semi-supervised learning is to learn a predictive function using a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning tries to combine these two sources of information in order to learn a predictor in an optimal way. Generally, labeled data is expensive while unlabeled data is ubiquitous, e.g. web pages. Hence, trying to exploit the distribution of unlabeled data during the training process is indeed highly interesting. Among popular semi-supervised algorithm are: Co-training, EM algorithms, Transductive inference,...(for a comprehensive survey of the topic see, e.g., Zhu, 2008; Zhu and Goldberg, 2009). Formally, the input space $\mathcal{X}$ can be split into two sets: the labeled points $\mathcal{X}_l = \{x_1, x_2, \ldots, x_l\}$ and the unlabeled points $\mathcal{X}_u = \{x_{l+1}, x_{l+2}, \ldots, x_{l+u}\}$. The labeled set $\mathcal{X}_l$ goes with its associated labels $\mathcal{Y}_l = \{y_1, y_2, \ldots, y_l\}$. A semi-supervised approach will try to predict the labels $\widehat{\mathcal{Y}}_u$ associated to $\mathcal{X}_u$. To achieve its goal the approach uses as input all the available input space $\mathcal{X}$. This makes the difference with supervised approaches that only use the labeled set $\mathcal{X}_l$ during the training. The intuition is that when lacking training data the distribution of the unlabeled points may be of great benefit.

Furthermore, we can split semi-supervised learning in two different settings: the *transductive* learning and the *inductive* learning. Transductive learning was introduced by Vapnik (1998). Its goal is to predict labels only on the test set used during the learning process. On the other hand, inductive learning has the objective to output a predictive function defined on the entire input space $\mathcal{X}$.

In recent years, graph-based semi-supervised learning has received a growing focus. A graph-based approach makes the hypothesis to work directly on a graph representation of the data. In such a setting, one tries to assign a label to the unlabeled nodes of a graph. Since the topology of the entire graph is used (including the unlabeled nodes), the problem is transductive. Indeed the models do not just use a subgraph formed by the labeled points, as in a supervised setting, but all the available graph topology. In

the case of a low labeling rate, i.e., when having the labels just for a small subset of $\mathcal{X}$, having all the graph topology may appear to be very helpful. More precisely, the problem is transductive since the models are designed to predict the labels of the unlabeled nodes already existing in the graph. In the remainder of this thesis, when speaking about semi-supervised classification we actually refer to transductive classification.

Contributions and Organization of this Chapter

This chapter presents one of the main contributions of this thesis:

- Experimental comparisons with various kernels on a graph (computed on eight different databases) show that the SoP correlation kernel obtains competitive performances in semi-supervised classification tasks.

This contribution has been published in the following journal paper:

Amin Mantrach, Luh Yen, Jerome Callut, Kevin Francoisse, Masashi Shimbo, and Marco Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1112Ð1126, 2010. ISSN 0162-8828

The subsequent part, Section 5.1, introduces basic assumptions made by graph-based semi-supervised learning. Section 5.2 shows how we can define loss functions based on these assumptions and deduce the sum-of-similarities framework. Finally, experimental comparisons with several state-of-the-art kernels on a graph and our Sum-over-Paths covariance kernel are exposed on eight different databases in Section 5.3.

5.1 Assumptions

To be effective, semi-supervised learning algorithms rely one some assumptions. This is also the case for simple supervised learning making the assumption that two close points into the input space should have a corresponding output. The *smoothness assumption* says that if two points are close in a high-density region so should be the corresponding outputs. On the other hand, if two points are close in a low-density region, then their outputs need not to be close. A special case is the *cluster* assumption stipulating that points lying in the same cluster are likely to be of the same class. This last assumption can be formulated as the following: the decision boundary should lie in a low-density region. Finally, a different but related assumption is the *manifold assumption* which states that the high-dimensional data lie approximately on a

low-dimensional manifold. The latter allows the learning algorithm to run in a corresponding low-dimensional space, hence, avoiding the curse of the dimensionality problem.

Zhou et al. (2003) illustrate these assumptions reported in Figure 5.1. The Figure shows two moons representing two classes. On the top, the red (square marks) moon and at the bottom, the blue (downside triangle marks). Given as inputs only one labeled sample per class (see Figure 5.1, a) we want to retrieve the entire two moons showed in Figure 5.1, d, down right subfigure. The subfigures (b) and (c) show the results obtained by a simple 1-NN and a SVM RBF Kernel. Both algorithms verify the smoothness assumption, i.e. two points that are close enough in the input space should have corresponding close outputs. Having two initial labeled points, one per class, as depicted on subfigure (a), a 1-NN will misclassify a large part of the red and the blue moons. This error derives simply by assigning to a new point the nearest neighbor label. It leads to propagating erroneously red labels (and in the opposite direction blue labels) through the boundary separating the two moons. Therefore, a good semi-supervised algorithm should take into account of some global constraints respecting the underlying manifolds and the cluster assumption. In the same way, a simple supervised SVM only trained on the same two initial labeled points infers a decision boundary that crosses high density regions. This is because it does not take into account of the existing unlabeled points in that region. Hence, we say that a naive SVM breaks the cluster assumption. In contrast, a semi-supervised approach will take as input the two initial labeled points, but also the distribution of the unlabeled points. Therefore, when inferring a decision boundary, a semi-supervised approach will take into account of high density regions as well. In other words, unlabeled point give an important information to predictive models.

5.2 Regularization framework

In this section, we introduce a multi-class semi-supervised framework. We propose to define a semi-supervised loss function respecting the local assumption. Indeed, following Zhou et al. (2003); Belkin et al. (2004a); Zhu et al. (2003), we consider an initial labeling for class c denoted by the column vector $\mathbf{y}^c$ which is an indicator vector containing as entries 1 for nodes belonging to class c and 0 otherwise. In the subsequent, when possible and for a better readability $\mathbf{y}^c$ will be shortened to $\mathbf{y}$. Then, let $\mathbf{f}$ be a fitness score vector of size $|V|$, where each node i ($f_i = \mathbf{e}_i^T\mathbf{f}$) contains the affinity of this node with the considered class. We are looking for the score labeling $\widehat{\mathbf{f}}$ that minimizes a least square error function Q defined on all the graph $\mathcal{G}$:

$$Q(\mathbf{f}) = \sum_{i \in \mathcal{G}} (f_i - y_i)^2 \tag{5.1}$$

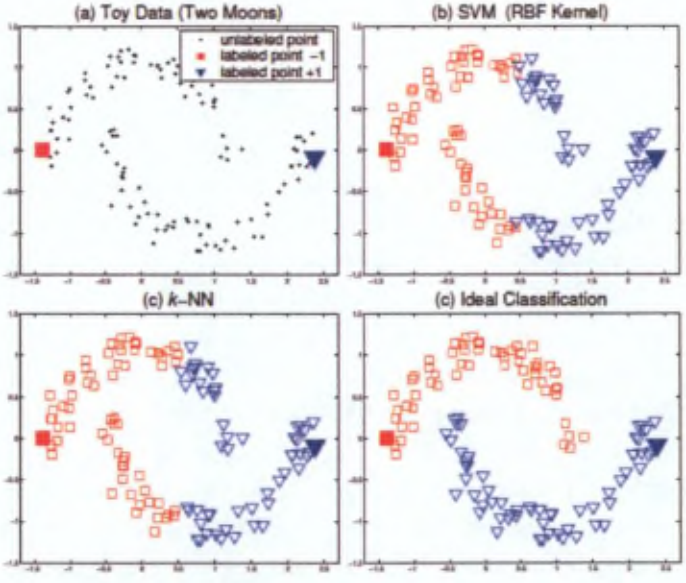


Figure 5.1: **Classification on the two moons pattern. (a) toy data set with two labeled points; (b) classifying result given by the SVM with a RBF kernel; (c) k-NN with k = 1; (d) ideal classification** Zhou et al. (2003)

Normally, this least square should be defined only on the training set. However, the framework of Zhou et al. (2003) considers that when facing unlabeled points the function $\widehat{f}$ should be also close to 0.

We add a regularization term suggesting that nearby points should have the same labeling on all nodes of the graph — labeled and unlabeled points (i.e. local assumption)

$$\sum_{i \in \mathcal{G}} a_{ij} \left(f_i - f_j\right)^2 \tag{5.2}$$

We rewrite this sum as

$$\sum_{i,j \in \mathcal{G}} a_{ij} \left(f_i - fj\right)^2 = \sum_{i,j \in \mathcal{G}} a_{ij} \left(f_i^2 - 2 f_i f_j + f_j^2\right) \tag{5.3}$$

$$\tag{5.4}$$

If the graph is undirected (i.e. a symmetric adjacency matrix), we have :

$$\sum_{i,j \in \mathcal{G}} a_{ij} f_i^2 = \sum_{i,j \in \mathcal{G}} a_{ij} f_j^2 \tag{5.5}$$

Hence, Equation (5.3) can be reformulated as

$$\sum_{i,j\in\mathcal{G}} a_{ij}\,(f_i - fj)^2 = 2\sum_{i,\in\mathcal{G}} f_i^2 \sum_{j,\in\mathcal{G}} a_{ij} - 2\sum_{i,j\in\mathcal{G}} a_{ij} f_i fj \tag{5.6}$$

$$= 2\mathbf{f}^{\mathrm{T}}(\mathbf{D}-\mathbf{A})\mathbf{f} \tag{5.7}$$

$$= 2\mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f} \tag{5.8}$$

with $\mathbf{D} = \mathbf{Diag}(\mathbf{A}^{\mathrm{T}}\mathbf{e})$ the diagonal matrix containing the degree of the nodes and $\mathbf{L} = \mathbf{D} - \mathbf{A}$ the un-normalized laplacian matrix. This laplacian regularization term means that fast changes in the predicted scores between close points (according to the affinity matrix $\mathbf{A}$) are penalized.

Thus, we are looking for the score fitness matrix $\widehat{\mathbf{f}}$ that minimizes the function Q. So, the first order derivate is

$$\partial_{\mathbf{f}}(Q) = \partial_{\mathbf{f}}\left((\mathbf{f}-\mathbf{y})^{\mathrm{T}}(\mathbf{f}-\mathbf{y}) + 2\mu\mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f}\right) \tag{5.9}$$

We may also introduce a normalization factor to reduce the intrinsic importance of popular nodes (high degree) by normalizing the contribution of each node by its degree. Then,

$$\partial_{\mathbf{f}}(Q) = \partial_{\mathbf{f}}\left((\mathbf{f}-\mathbf{y})^{\mathrm{T}}(\mathbf{f}-\mathbf{y}) + 2\mu\mathbf{D}^{-\frac{1}{2}}\mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{f}\right) \tag{5.10}$$

$$= 2(\mathbf{f}-\mathbf{y}) + 2\mu\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}\mathbf{f} \tag{5.11}$$

The second order derivative is

$$\frac{\partial(Q)}{\partial\mathbf{f}\partial\mathbf{f}^{\mathrm{T}}} = 2\mathbf{I} + 2\mu\mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} \tag{5.12}$$

which is a positive definite matrix (for $\mu > 0$), ensuring that $\partial_{\mathbf{f}}(Q)$ is minimized when set to 0.

By setting the first order derivative $\partial_{\mathbf{f}}(Q)$ to zero we get

$$\widehat{\mathbf{f}} = (\mathbf{I} + \mu\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2})^{-1}\mathbf{y} \tag{5.13}$$

$$= \beta(\mathbf{I} - \alpha\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^{-1}\mathbf{y} \tag{5.14}$$

$$= \beta\mathbf{K}_{\mathrm{NRL}}\mathbf{y} \tag{5.15}$$

with $\alpha = \frac{\mu}{1+\mu}$ and $\beta = \frac{1}{1+\mu}$, thus $\alpha + \beta = 1$.

NRL states for normalized regularized laplacian. Indeed, regularized holds for the μ regularization factor derived directly from the loss function Q (see Equation (5.10)), and normalized holds for the $\mathbf{D}^{-1/2}$ normalization. When using the derived product

$\mathbf{K}_{\text{NRL}}\mathbf{y}^c$ for classification purpose, we assign to each node the class having the maximum score value, i.e. $\arg\max_{c \in \mathcal{L}} \mathbf{K}_{\text{NRL}}\mathbf{y}^c$, where $\mathcal{L}$ is the set of labels. Hence, we may ignore the β term when the sum-of-similarities is computed (i.e. the product between the kernel matrix and the $\mathbf{y}$ class indicator vector). The laplacian is a direct consequence of the semi-supervised *smoothness* constraint suggesting that nearby points should have the same labeling, which in case of undirected graphs is formulated with the laplacian matrix (see the development from Equation(5.8) to Equation(5.2)). Equation(5.8) also expresses a global constraints since the local consistency principle simultaneously holds on all the nodes of the graph.

The product between the kernel matrix and the initial labeling vector $\mathbf{y}$ will be referred in the remainder of this thesis as the sum-of-similarities. Indeed, computing this product results in a score vector which measures for each node the sum of its similarity with all the other nodes of a specified class c, indicated by $\mathbf{y}$. Considering the sum instead of the mean allows taking into account the prior distribution of the different classes. Indeed, the more represented classes have more 1 entries in their corresponding $\mathbf{y}$ indicator vector and therefore contribute more in the sum-of-similarities than under-represented classes. We may note that the computation of the kernel matrix $\mathbf{K}_{\text{NRL}}$ is completely unsupervised since it does not depend on original labels but solely on the original adjacency matrix $\mathbf{A}$. This observation implies that the way labels are diffused in the network entirely depends on the graph topology. Hence, this sum-of-similarities is said to be a *framework* since we may plug-in other kernels to target the same semi-supervised classification task.

Other loss functions may be chosen in order to derive other interesting kernels for semi-supervised classification based on the sum-of-similarities principles. For instance, instead of normalizing the regularized laplacian as suggested by Equation (5.12), we may normalize the mean square loss (see Equation(5.2)) having the following loss function Q' defined on the labeled training set $\mathcal{L}$:

$$Q'(\mathbf{f}') = \sum_{i \in \mathcal{G}} (d_{ii} fi - y_i)^2 \tag{5.16}$$

Minimizing Q' means we are looking for the normalized vector $\widehat{\mathbf{f}}' = \mathbf{D}^{-1}\mathbf{f}$ that minimizes the function Q'. Adding the same un-normalized regularization term we want to minimize

$$\partial_{\mathbf{f}'}(Q') = \partial_{\mathbf{f}'}\left(\sum_{i \in \mathcal{L}} ||\mathbf{e}_i^{\mathrm{T}}(\mathbf{D}\mathbf{f}' - \mathbf{y})||^2 + 2\mu \mathbf{f}'^{\mathrm{T}}\mathbf{L}\mathbf{f}'\right) \tag{5.17}$$

$$= 2(\mathbf{D}\mathbf{f}' - \mathbf{y} + \mu\mathbf{L}\mathbf{f}') \tag{5.18}$$

Setting this derivate to zero we obtain

$$\widehat{\mathbf{f}}' = (\mathbf{D} + \mu\mathbf{L})^{-1}\mathbf{y} \tag{5.19}$$

By posing, again, $\alpha = \frac{\mu}{1+\mu}$ and $\beta = \frac{1}{1+\mu}$, thus $\alpha + \beta = 1$, we obtain

$$\begin{aligned}
\widehat{\mathbf{f}}' &= \beta((1-\alpha)\mathbf{D} + \alpha\mathbf{L})^{-1}\mathbf{y} && (5.20)\\
&= \beta(\mathbf{D} - \alpha(\mathbf{D} - \mathbf{L}))^{-1}\mathbf{y} && (5.21)\\
&= \beta(\mathbf{D} - \alpha\mathbf{A})^{-1}\mathbf{y} && (5.22)\\
&= \beta\mathbf{K}_{\text{NRWR}}\mathbf{y} && (5.23)
\end{aligned}$$

These calculations result in a normalized version of the random walk with restart kernel introduced by Pan et al. (2006), initially inspired by the famous PageRank algorithm (see, e.g., Brin and Page, 1998; Page et al., 1999). It also corresponds to the regularized commute-time kernel since the inner product defined by this Gram matrix measures the commute time (see, e.g., Ham et al., 2004; Fouss et al., 2007b). By defining other loss functions we may derive other kernels, e.g. the regularized laplacian kernel.

A number of popular algorithms — such as support vector machines (SVMs), ridge regression – may be broadly interpreted as regularization algorithms in an appropriately chosen reproducing Hilbert space. Belkin et al. (2004b, 2005); Sindhwani et al. (2005) proposed to extend an established framework for function learning in reproducing kernel Hilbert spaces (RKHS). By adding a graph laplacian regularization term to the traditional cost functions of SVMs or ridge regression, we obtain a semi-supervised version of these algorithms: LapSVM (Laplacian SVM) and LapRLS (Laplacian RLS). Usually, the laplacian regularization corresponds to smoothing the boundary decision in order to avoid boundaries in a high density region.

Wang et al. (2008) compared the different existing regularization techniques while proposing to extend some limitations of the sum-of-similarities regularization framework. According to their tests, there is no significant difference in terms of classification rate between the LapSVM, LapRLS and the sum-of-similarities framework. Therefore, we will restrict our study to the sum-of-similarities framework applied with different graph-kernels. Furthermore, the LapSVM demands to first solve a quadratic program. Moreover, in Chapter 6, it will be shown that the sum-of-similarities framework may often be solved by an iterative algorithm in order to reduce the computation time and apply it to large-scale networks. In this case, iterative algorithms may be seen as diffusing labels into the network. We will also show that such label propagation based algorithms have a intuitive interpretation in terms of random walks into the network.

It may also be interesting to use this framework for kernels not deduced from a lapla-

Category	Size
High-revenue	572
Low-revenue	597
Total	1169
Majority class proportion	51.07%

Table 5.1a: Class distribution for the `IMDb-proco` data set.

cian regularization. In other words, we want to assess empirically the performance that other kernels on a graph (i.e. not directly derived from a loss function defined from a semi-supervised classification problem) may obtain and compare their respective performance. Hence, the next Section will investigate the performance of various kernels obtained by this sum-of-similarities framework for state-of-the art graph kernels on various medium size networks (i.e. between $1K$ and $5K$ nodes) data sets. More specifically, we want to test how well the SoP covariance kernel deduced from the randomized shortest path framework presented in Chapter 3 behaves for a semi-supervised task.

5.3 Experiments

In this experiment, we address the task of classification of unlabeled nodes in partially labeled graphs. Notice that the goal of this experiment is not to design a state-of-the-art semi-supervised classifier; rather it is to study the performances of the proposed SoP correlation measure, in comparison with other kernels on a graph.

Data sets. The different classification models, referred to as classifiers, are compared on eight data sets that have been used previously for semi-supervised classification: the four universities `WebKB cocite` data sets (see Zhou et al., 2005; Macskassy and Provost, 2007), the two `industry` data sets (see Macskassy and Provost, 2007), the `IMDb prodco` data set (see Macskassy and Provost, 2007) and the `CoRA cite` data set (see Macskassy and Provost, 2007)*.

IMDb: The collaborative Internet Movie Database (`IMDb`, Macskassy and Provost (2007)) has several applications such as making movie recommendation or movie category classification. The classification problem focuses on the prediction of the movie notoriety (whether the movie is a box-office or not). It contains a graph of movies linked together whenever they share the same production company. The weight of an edge in the resulting graph is the number of production companies two movies have in common. The `IMDb-proco` graph contains 1169 movies which have the class distribution as shown in Table 5.1a.

*These data sets (original source) can be downloaded from `http://netkit-srl.sourceforge.net/data.html`. The preprocessed version in Matlab/Octave format used in our experiments is available from `http://iridia.ulb.ac.be/~amantrac/pub/SoP_TPAMI.zip`.

Category	Size	
	industry-yh	industry-pr
Basic Materials	104	83
Capital Goos	83	78
Conglomerates	14	13
Consumer Cyclical	99	94
Consumer NonCyclical	60	59
Energy	71	112
Financial	170	268
Heathcare	180	279
Services	444	478
Technology	505	609
Transportation	38	47
Utilities	30	69
Total	1798	2189
Majority class proportion	28.1%	27.8%

Table 5.1b: Class distribution for the `industry-yh` and `industry-pr` data sets.

Category	Size
Case-based	402
Genetic Algorithms	551
Neural Networks	1064
Probabilistic Methods	529
Reinforcment Learning	335
Rule Learning	230
Theory	472
Total	3583
Majority class proportion	29.70%

Table 5.1c: Class distribution for the `CoRA cite` data set.

Category	Size			
	Cornell	Texas	Washington	Wisconsin
Course	54	51	170	83
Department	25	36	20	37
Faculty	62	50	44	37
Project	54	28	39	25
Staff	6	6	10	11
Student	145	163	151	155
Total	346	334	434	348
Majority class proportion	41.9%	48.8%	39.2%	44.5%

Table 5.1d: Class distribution for the `WebKB cocite` data set.

Industry: `Industry` regroups two datasets Macskassy and Provost (2007). The `industry-pr` data set is based on 35,318 Newswire press releases. The companies mentioned in each press release were extracted and an edge was placed between two companies if they appeared together in a press release. The `industry-yh` data set is based on 22,170 business news stories collected from the web. An edge between two companies is placed if they appeared together in a story. The weight of an edge is the number of such cooccurrences found in the complete corpus. To classify a company, Yahoo!'s 12 industry sectors have been used in the two industry data sets. The details about the two industry datasets are reported in Table 5.1b.

CoRA: `CoRA cite` is a graph of 3,583 nodes collected from machine learning research papers labeled into 7 different topics (see Macskassy and Provost, 2007). Papers are linked if they share a common author, or if one cites the other. The composition of the `CoRA cite` data set is reported in Table 5.1c.

WebKB: `WebKB` consists of sets of web pages gathered from four computer science departments (one for each university), with each page manually labeled into 6 categories: course, department, faculty, projet, staff, and student (see Macskassy and Provost, 2007). Two pages are linked by co-citation (if x links to z and y links to z, then x and y are co-citing z). The composition of the data set is shown in Table 5.1d.

Classification models. The compared classifiers are based on (1) the SoP correlation kernel (SoP) introduced in this thesis, (2) the normalized commute-time (NCT) kernel (see, e.g., Zhou et al., 2005), (3) the randomized shortest-path similarity (RSP) induced by the randomized shortest-path dissimilarity (see Yen et al., 2008), (4) the diffusion map kernel (DM) (see, e.g., Fouss et al., 2006) computed from the diffusion distance (see, e.g., Nadler et al., 2005, 2006; Pons and Latapy, 2005, 2006), (5) the commute-time (CT) kernel (see, e.g., Fouss et al., 2007a; Saerens et al., 2004), (6) the regularized laplacian (RL) kernel (see, e.g., Chebotarev and Shamis, 1997, 1998), (7) the Netkit (Netkit) framework (see Macskassy and Provost, 2007) and (8) a simple k nearest neighbor (KNN). The first six classifiers are kernel-based while the two last are included as baselines.

The RSP dissimilarity, depends on a relative entropy parameter θ and has the interesting property of reducing, on one end, to the standard shortest-path distance when θ is large and, on the other end, to the commute-time (or resistance) distance when θ is small (near zero). Intuitively, it corresponds to the expected cost incurred by a random walker in order to reach a destination node from a starting node while maintaining a constant relative entropy (related to θ) spread in the graph. The corresponding natural kernel matrix based on the squared dissimilarities matrix $\mathbf{D}_{\text{RSP}}$ is derived in the standard way by $\mathbf{K}_{\text{RSP}} = -\frac{1}{2}\mathbf{H}\mathbf{D}_{\text{RSP}}\mathbf{H}$, where $\mathbf{H} = \mathbf{I} - \mathbf{e}\mathbf{e}^{\mathrm{T}}/n$ is the centering matrix, $\mathbf{e}$ is a column vector full of ones and $\mathbf{D}_{\text{RSP}}$ is a matrix containing the symmetric squared dissimilarities (see Yen et al., 2008). This is the standard way for deriving a similarity from a dissimilarity (see, e.g., Borg and Groenen, 1997) when the dissimilarity matrix

contains squared distances.

For the kernel-based methods (SoP, NCT, RSP, DM, CT, RL), the unlabeled nodes classification was performed according to a simple sum of the similarities with the labeled nodes (as for instance described in Zhou et al., 2005). More precisely, let us define an n-dimensional indicator vector, $\mathbf{y}^c$, containing as entries a "1" when the corresponding node belongs to class c and "0" otherwise (in which case the node is unlabeled or belongs to another class). For each node, its similarity with nodes belonging to class c is contained in the column vector $\mathbf{K}\mathbf{y}^c$ where $\mathbf{K}$ is the graph kernel matrix. Then, each node is assigned to the class showing the largest similarity; the predicted class index is thus provided by $\text{argmax}_c(\mathbf{K}\mathbf{y}^c)$ for all nodes.

Note that for the NetKit classifier (Netkit), we only tested the default parameters present in the framework, which generally provide good results (see Macskassy and Provost, 2007). This method therefore defines baseline performances on each data set. We also report as baseline the results of a KNN classifier. Our implementation of the KNN consists in taking all neighbors at maximum k steps of the considered node. An unlabeled node will be labeled with the label that is most represented in the set of nodes located at maximum k steps. Each vote is weighted by the similarity in terms of number of steps ($1/k$) with the node of interest.

Experimental methodology. The classification accuracy will be reported for several labeling rates (20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The labels of remaining nodes are removed and used as test data. For each considered labeling rate, 100 random node label deletions (100 runs) were performed, on which performances are averaged. For each unlabeled node, the various classifiers predict the most suitable category according to the procedures described in the previous paragraph.

For each run, a 5-fold internal cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (for instance, the parameter θ for the SoP correlation). Thus, the performance on each run is assessed on the remaining unlabeled nodes with the hyper-parameter tuned during the cross-validation. We report, for each method and each labeling rate, the average classification rate obtained on the 100 runs.

Results and discussion. Comparative results for each method are reported on the eight different data sets in Figure 5.3, 5.4, 5.5 and 5.6. Clearly the SoP, the RSP and the NCT kernel-based classifiers outperform the other approaches on the majority of data sets. The RSP obtains the best results on two data sets (`WebKB-washington` and `WebKB-texas`) while the NCT kernel achieves the best performances on the `WebKB-winsonsin` data set. Notice that the results of the RSP kernel are not reported on the dataset `industries` and `CoRA` because of prohibitive computation time. The NetKit package provides poor results except for the `industry` data sets where it achieves the best results. Notice that the DM kernel and the RL kernel are

competitive methods but never outperform the three leading methods on the tested data sets. In the case of an undirected graph, the NCT kernel differs from the RL method only by the normalization of the laplacian matrix. This normalization clearly boosts the performance for the classification task. The KNN obtains good results on the `IMDb` data set (Figure 5.6) and is based on the hypothesis that propagating the labels from nodes to nodes is a good way to label unlabeled nodes. Intuitively, this measure will be ineffective in case of a low labeling rate. This can indeed be observed on all data sets and more strongly on the `IMDb` and `CoRA` data sets (Figure 5.6).
The SoP measure is based on the same diffusing hypothesis. The way a node is correlated to the others is tuned through the θ parameter. A high θ value means that we consider only shortest paths. Considering only shortest paths gives small chance to two nodes to appear together on the same path, which leads to a constantly low correlation. This can be observed on Figure 4.5(c) where we display the SoP matrix obtained for the graph of Figure 4.1. The correlation between nodes is low and quite the same between all pair of nodes so that discriminating between nodes is difficult. Inversely, a θ value close to 0 means that we consider all paths. Considering all paths leads to a constantly high correlation. This can be observed on Figure 4.5(a) where we display the SoP matrix obtained for the graph of Figure 4.1. The correlation between nodes is high and quite the same between all pair of nodes so that discriminating between nodes is difficult. To cluster better, we need another θ value of, for example 0.1, which will give more weight to close nodes and improve the diffusion of unlabeled nodes. The Figure 4.5(b) shows that for a θ of 0.1 the left hand-side nodes are less correlated to right-hand side nodes and hence clustering nodes. This is confirmed by the experiments shown in Figure 5.2 where we observe the influence of the θ parameter on the classification rate on the `WebKB-washington` data set for increasing labeling rates. On this data set, the best θ parameter is located around 0.2 and 0.8 according to the labeling rate considered.

5.4 Related work

Instead of learning a general predictive function, Vapnik (1998) proposed the transductive learning setting where predictions are only made on pre-specified number of unlabeled test points. Indeed, this allows the learning scheme to exploit the location of the unlabeled points for inferring a decision boundary. In this spirit, transductive support vector machines (TSVMs) include test points in the computation of the margin. However, TSVM optimization problems can be viewed as a mixed-integer problem with a quadratic objective and linear constraints. Unfortunately, currently, no algorithm is known to efficiently find a global optimal solution to this optimization problem.
This led to other formulations of transductive learning algorithms: graph and spectral partioning (see e.g. Zhou et al., 2003; Belkin et al., 2004a; Zhu et al., 2003; Belkin

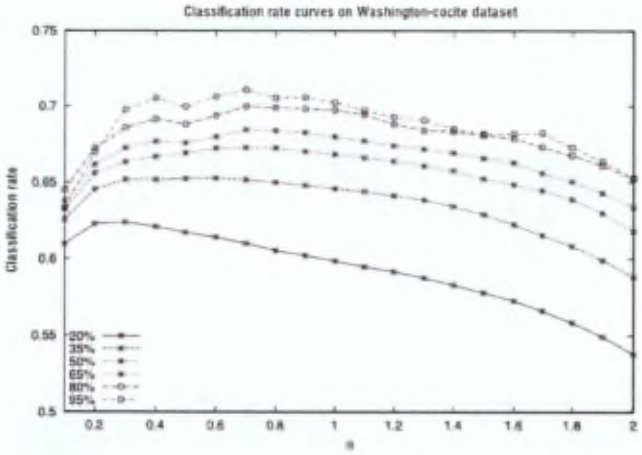
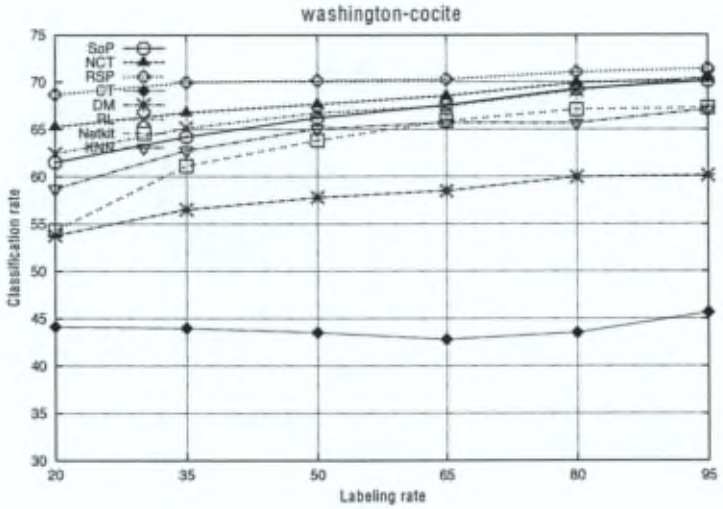


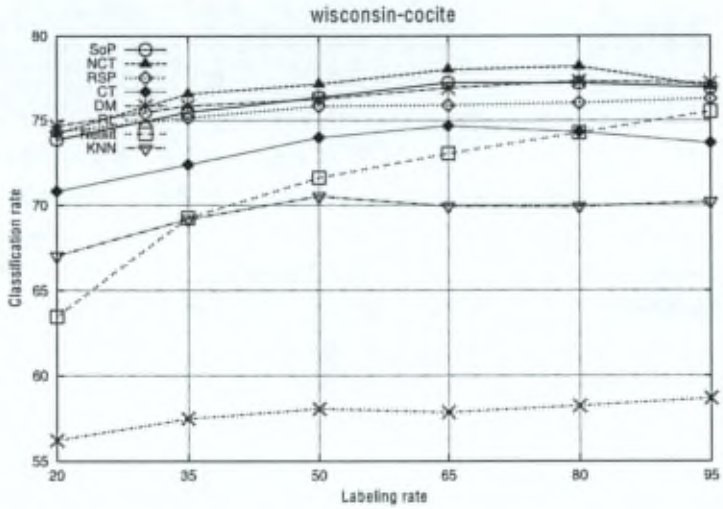
Figure 5.2: **Curves of the classification rates obtained for different values of the parameter θ (x-axis) and different labeling rates.**

et al., 2004b, 2005; Sindhwani et al., 2005). Their main objective is to exploit the relationship between the geometry of unlabeled points and their labels, but this time, without the computational inconvenient properties of the TSVMs. In this topic, various graph-based algorithms have recently been proposed (see e.g. Zhou et al., 2003; Belkin et al., 2004a; Zhu et al., 2003). They rely on the idea of building a graph where nodes are data points and edges are affinities or costs between nodes. Known labels are diffused into the graph in order to label unlabeled nodes (i.e. label propagation algorithms). These algorithms share a common framework where one minimizes a quadratic criterion. The great advantage of this approach lies in the fact that it can be solved by a linear system and thus may be applied on large-scale network, as we investigate in detail in Chapter 6. Belkin et al. (2004b, 2005); Sindhwani et al. (2005) also exploit the geometry of unlabeled points but with another related framework. They propose to extend an established framework for function learning in reproducing kernel Hilbert spaces (RKHS). This leads to the natural semi-supervised extension of the traditional SVM and ridge regression: LapSVM (Laplacian SVM) and LapRLS (Laplacian RLS). Since they require both solving a quadratic program, these techniques do not scale on large-scale network.

More recently, Wang et al. (2008) compared the different existing regularization techniques while proposing to overcome some limitations of the sum-of-similaritites regularization framework of Zhou et al. (2003); Belkin et al. (2004a). Their algorithm defines a joint iterative optimization over the classification function and a balanced label matrix. Each minimization step requires $\mathcal{O}(n^2)$ time and the total runtime of the greedy algorithm is in $\mathcal{O}(n^3)$. According to their tests there is no significant difference in term of classification rate between the LapSVM, LapRLS and the sum-of-similarities



(a)



(b)

Figure 5.3: **Classification rates in percent, averaged over 100 runs, obtained on partially labeled graphs, for an increasing labeling rate of 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP (sum-over-paths) kernel, the NCT (normalized commute-time) kernel, the RSP (randomized shortest-path) kernel, the CT (commute-time) kernel, the DM (diffusion map) kernel, the RL (regularized laplacian) kernel, the Netkit (NetKit) framework and the KNN algorithm. The graphs show the results obtained on the `washington` and `wisconsin` datasets.**

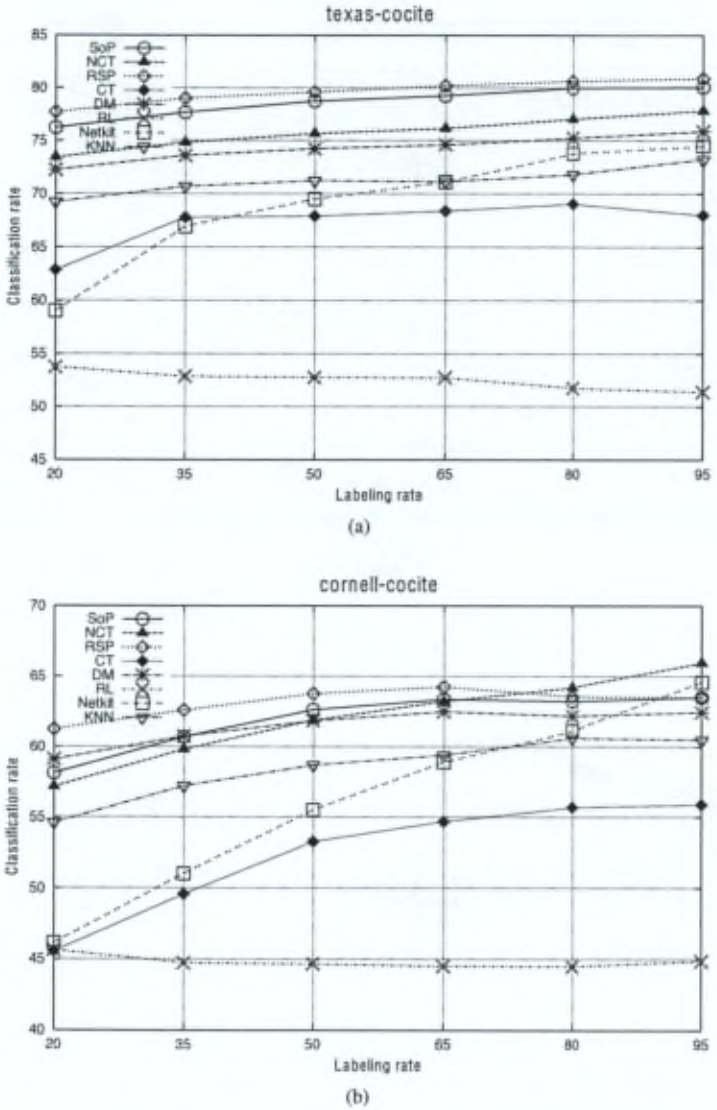


Figure 5.4: **Classification rates in percent, averaged over 100 runs, obtained on partially labeled graphs, for an increasing labeling rate of 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP (sum-over-paths) kernel, the NCT (normalized commute-time) kernel, the RSP (randomized shortest-path) kernel, the CT (commute-time) kernel, the DM (diffusion map) kernel, the RL (regularized laplacian) kernel, the Netkit (NetKit) framework and the KNN algorithm. The graphs show the results obtained on the `texas` and `cornell` `WebKB` data sets.**

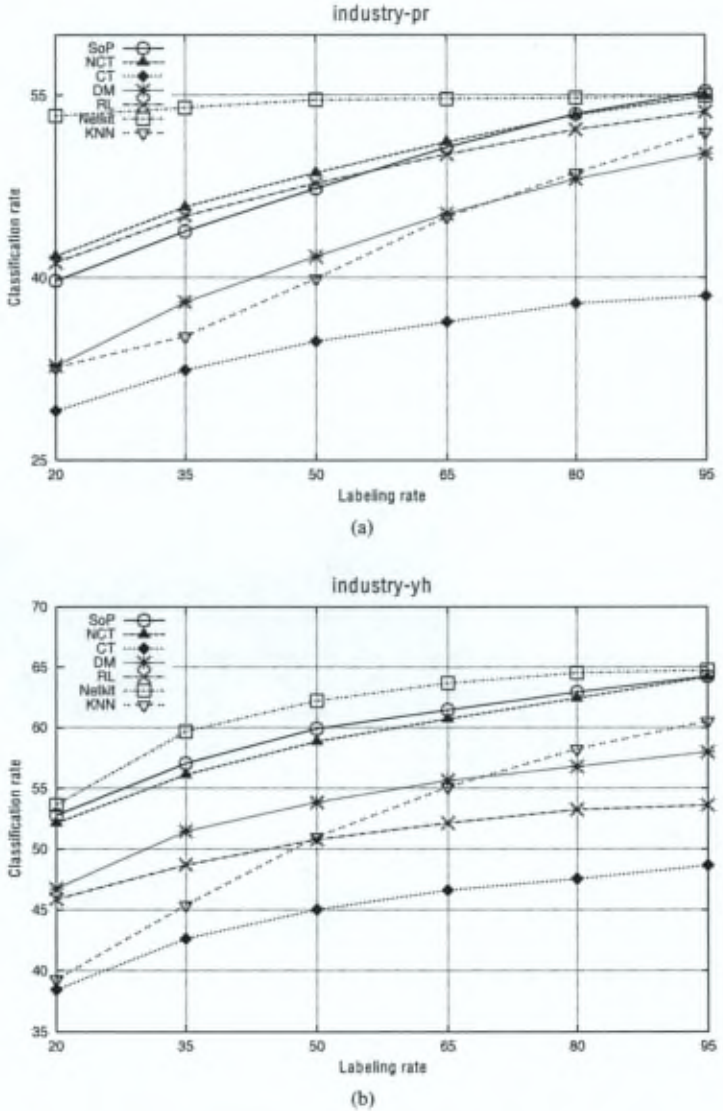


Figure 5.5: **Classification rates in percent, averaged over 100 runs, obtained on partially labeled graphs, for an increasing labeling rate of 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP (sum-over-paths) kernel, the NCT (normalized commute-time) kernel, the CT (commute-time) kernel, the DM (diffusion map) kernel, the RL (regularized laplacian) kernel, the Netkit (NetKit) framework and the KNN algorithm. Notice that the results of the RSP kernel are not reported on the `industries` data sets because of prohibitive computation time. The graphs show the results obtained on the two `industries` datasets.**

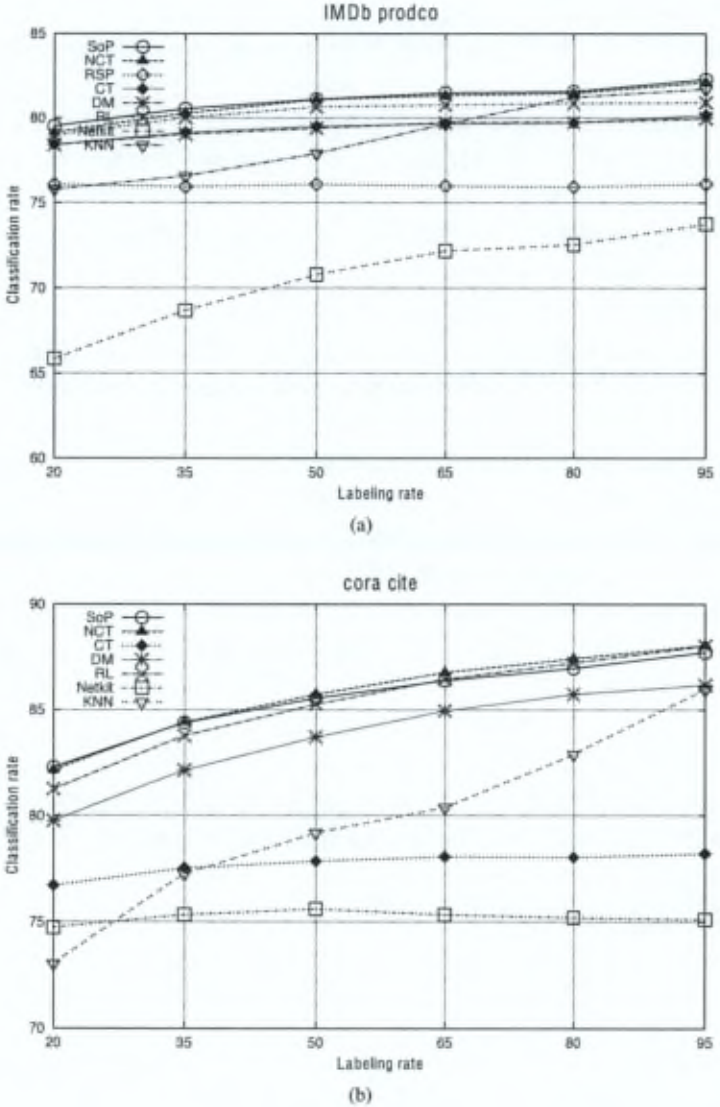


Figure 5.6: **Classification rates in percent, averaged over 100 runs, obtained on partially labeled graphs, for an increasing labeling rate of 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP (sum-over-paths) kernel, the NCT (normalized commute-time) kernel, the RSP (randomized shortest-path) kernel, the CT (commute-time) kernel, the DM (diffusion map) kernel, the RL (regularized laplacian) kernel, the Netkit (NetKit) framework and the KNN algorithm. The graphs show the results obtained on the IMDb and CoRA data sets Macskassy and Provost (2007). Notice that the results of the RSP kernel are not reported on the dataset CoRA) because of prohibitive computation time.**

framework, whereas the performance of their method is significantly better than others on various data sets.

A complete benchmark analysis of the different semi-supervised learning algorithms has been published in Chapelle et al. (2006, chapter 21, p. 377). The main conclusion is that no algorithm is uniformly better than the others. One has to distinguish between cluster-based algorithms (TSVMs, i.e. learning a boundary decision) and manifold-based algorithms (graph-based) (see e.g. Zhou et al., 2003; Belkin et al., 2004a; Zhu et al., 2003; Belkin et al., 2004b, 2005; Sindhwani et al., 2005). There is no "black box" solution and a good understanding of the nature of the data (more cluster-like data or more manifold-like data) is required to perform successful semi-supervised learning. As suggested by Corduneanu (2006), powerful semi-supervised learning distinguishes itself through the ability to make use of prior available knowledge about the domain and data distribution in order to relate data and labels and improve classification.

Readers interested in further readings about semi-supervised learning may fruitfully turn to Chapelle et al.'s book (2006) on state-of-the-art in semi-supervised learning as well as the online survey held by Zhu (2008); Zhu and Goldberg (2009).

The same problem of within-network classification has been tackled by the statistical learning community (see Getoor and Taskar, 2007, for a survey of the field). The main principle consists in using a relational classifier into a collective inference procedure. A relational classifier generally estimates node class probabilities using the direct neighborhood. Among the simplest classifier: the weighted-vote relational classifier which estimates class-membership as the weighted mean of the class-membership probabilities of the neighbor nodes (see Macskassy and Provost, 2003). See Macskassy and Provost (2007) for a non-exhaustive list and algorithms description of more sophisticated node relational classifiers. A collective inference method has the objective to estimate the class-membership of all the nodes by propagating known information through the network. This idea has been used successfully in iterative classification (IC)(see Neville and Jensen, 2000), relaxation labeling (see Chakrabarti et al., 1998) and belief propagation (see Murphy et al., 1999). All these methods make use internally of the relational classifier. For instance iterative classification repeatedly classifies nodes using a relational classifier doing prediction on the current state of the graph until no nodes change their label. More recently, Maes et al. (2009) propose a new family of methods: Simulated IC. These methods help to reduce the training bias of standard IC methods by simulating inference during learning.

Chapter 6

Application to Large Sparse Graphs

Despite the growing need for dealing with huge real-world networks, few of the existing semi-supervised learning methods scale up to large graphs so that semi-supervised classification on large graphs has become one of the current central issues; see the survey Zhu (2008, Section 6.3). Indeed, the techniques that scale well (see, e.g., Macskassy and Provost, 2007) are not always competitive when compared to state-of-the-art graph-based metrics (Fouss et al., 2007b) such as the regularized Laplacian kernel (Chebotarev and Shamis, 2002), the sum-over-paths (SoP) covariance (see Section 4.2), the random walk with restart similarity and its normalized version (see, e.g. Pan et al., 2006; Tong et al., 2007; Fouss et al., 2007b), or the Markov diffusion kernel Fouss et al. (2006). A naive application of these graph kernel-based approaches does not scale well since it relies on the computation of a dense similarity matrix, which usually requires a matrix inversion. Techniques approximating the inverse of the matrix usually require some strong properties on the matrix, like the positive semi-definiteness (Bach and Jordan, 2005), and are only conceivable for medium-size graphs (up to 50,000 nodes) – for larger graphs, a dense similarity matrix cannot be computed and stored into memory.
This chapter tackles this problem with two different approaches. The first approach is based on existing, competitive, kernels on a graph, but it explicitly avoids the computation of the pairwise similarities between the nodes (following an idea suggested by Zhou et al. (2003, 2005)). Indeed, as opposed to Tong et al. (2007, 2008), Zhou et al. suggest to avoid computing each pairwise measure and solving a system of linear equations instead. We design two iterative algorithms along this approach, each based on a different state-of-the-art similarity metric: the SoP covariance kernel and the normalized random walk with restart. This kernel on a graph was called regularized commute-time kernel by Fouss et al. (2007b) and is closely related to the modified Laplacian matrix (Ito et al., 2005) and the random walk with restart similarity (see Sec-

tion 4.1.4). As suggested initially by Zhou et al. (2003, 2005), the algorithms directly approximate the sum of similarities to labeled nodes. The second approach takes its inspiration from the randomized shortest path framework of Saerens et al. (2009) and the $\mathcal{D}$-walk algorithm based on bounded random walks of Callut et al. (2008). In this case, a random walk betweenness, measuring how well a node is "in-between" each class, is derived from a lattice structure constructed from a biased random walk on the graph.

Contribution and Organization of this Chapter

This part makes three main contributions:

- It provides three algorithms to address within-network semi-supervised classification tasks on large, sparse, directed graphs. All these algorithms have a **computing time linear in the number of edges** of the graph. Moreover, an algorithm allowing to compute the SoP betweenness centrality is also proposed.
- It validates the three proposed algorithms on eight medium-size standard data sets and compares them to state-of-the-art techniques. Their performances are shown to be competitive in comparison with the other techniques.
- It introduces a **novel benchmark data set**: the `U.S. patents` citation network, on which our three algorithms obtain competitive results. Results are also computed on a standard large scale data set introduced by Chapelle et al. (2006).

These contributions has been submitted for publication to the Pattern Recognition journal:

Amin Mantrach, Nicolas van Zeebroeck, Pascal Francq, Masashi Shimbo, Hugues Bersini and Marco Saerens. Semi-supervised Classification and Betweenness Computation on Large, Sparse, Directed Graphs, *submitted for publication to Pattern Recognition*, PR-D-09-01097R

In the subsequent part, Section 6.1, two iterative algorithms are derived from the assumption of consistency. Further, Section 6.2 defines a biased bounded betweenness and proposes a forward/backward algorithm to compute it. Section 6.3 applies our three algorithms to semi-supervised classification tasks and compares the results with various state-of-the-art techniques. A novel benchmark data set is also introduced: the `U.S. patents` citation network on which our three algorithms are assessed. In Section 6.4, a brief survey of the domain is commented. Finally, the last part of the chapter includes conclusions and remarks as well as further extensions.

6.1 Kernel-based Semi-Supervised Classification on Large Sparse Graphs

Three approaches for semi-supervised classification on large sparse graphs are introduced in this chapter. The first two approaches (detailed in subsections 6.1.3 and 6.1.4) are based on approximating or bounding standard kernel-based techniques. They will therefore be referred to as *approximate approaches*. The third approach, discussed in detail in Section 6.2, is a generalization of the discriminative random walks classifier ($\mathcal{D}$walks, Callut et al. (2008)).

6.1.1 Kernel-based Classification

The approximate approaches are kernel-based and adopt the simple following classification procedure (the consistency method), initially proposed by Zhou et al. (2003, 2005) (see also, e.g., Belkin et al., 2004a; Wang et al., 2008; Yajima and Kuo, 2006). Based on a regularization framework optimizing a loss, this classification procedure takes both available class labels and smoothness into account. The resulting decision procedure derived from a a simple sum of similarities (each similarity being provided by an element of the graph kernel matrix) with the labeled nodes Zhou et al. (as described in 2003, 2005, for instance). This technique has been used with other kernels than those initially proposed by Zhou et al. with competitive results (see the previous chapter). It corresponds to a simple alignment between the kernel matrix and the class membership vector.

More precisely, suppose that we are given a meaningful proximity matrix $\mathbf{K}$ (usually a graph kernel matrix) providing similarities k_{ij} between each pair of nodes of the graph $\mathcal{G}$. For each node, its similarity with nodes belonging to class c is contained in the column vector $\mathbf{s}^c = \mathbf{K}\mathbf{y}^c$. Then, each node is assigned to the class showing the largest similarity; the predicted class index is thus provided by $\text{argmax}_c(\mathbf{s}^c)$ for all nodes. We propose to directly estimate this sum of similarities $\mathbf{s}^c$ for two different metrics, i.e. the *sum-over-paths* (SoP) covariance (see Section 4.2) and the so-called *regularized commute-time kernel* (see, e.g., Fouss et al., 2007b), closely related to the modified Laplacian matrix (see, e.g., Ito et al., 2005) and the random-walk with restart (RWR) similarity measure (see Section 4.1.4), that have been shown to be competitive on such tasks (see, e.g., previous chapter, Fouss et al., 2007b; Zhou et al., 2005). Because of the widespread use and popularity of the RWR measure, the regularized commute-time kernel will be called the *normalized random walk with restart* kernel in this thesis. Notice that such iterative updates for semi-supervised classification have been seen earlier (see, e.g., Rao et al., 2008; Zhu et al., 2003; Zhou et al., 2003; Szummer and Jaakkola, 2001); for a good review of these techniques see Chapelle et al. (2006, Chapter 11). In the remainder, we propose to apply such iterative procedure to two novel graph kernels

— that have been shown to perform well previously on medium size graphs — not yet applied on large-scale networks . Finally, notice also that column i of the kernel matrix, $\mathbf{K}\mathbf{e}_i$ (the similarities from node i), can be approximated with the same method by using $\mathbf{e}_i$, i.e. column i of the identity matrix, instead of $\mathbf{y}^c$.

6.1.2 The Approximate Normalized Regularized Laplacian

Zhou and Scholkopf (2004); Zhou et al. (2005) suggested to avoid computing explicitly the normalized regularized laplacian kernel to apply the sum-of-similarites framework. Indeed,

$$\mathbf{K}_{\text{NRL}}\mathbf{y} = (\mathbf{I} - \alpha\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^{-1}\mathbf{y} \tag{6.1}$$

The matrix $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is similar to the matrix $\mathbf{P}$ and hence has the same eigenvalues. Since $\mathbf{P}$ is a stochastic matrix its eigenvalues are in $[-1, 1]$, and so the eigenvalue of $\alpha\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ are in $(-1, 1)$ (remember that $0 < \alpha < 1$).
It follows that, when $t \to \infty$, $\sum_{i=0}^{t}(\alpha\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^i \to (\mathbf{I} - \alpha\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^{-1}$.
Such that

$$\mathbf{K}_{\text{NRL}}\mathbf{y} = \Big(\sum_{i=0}^{t}(\alpha\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})^i\Big)\mathbf{y} = \widehat{\mathbf{s}}(\infty) \tag{6.2}$$

We obtain the following recurrence equation scheme for computing $\widehat{\mathbf{s}}(\tau)$:

$$\begin{cases} \tilde{\mathbf{s}}(0) \leftarrow \mathbf{y} \\ \tilde{\mathbf{s}}(\tau) \leftarrow \alpha\,\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\,\tilde{\mathbf{s}}(\tau - 1) + \tilde{\mathbf{s}}(0), \quad \text{for } t = 1 \ldots \tau \end{cases} \tag{6.3}$$

$\tilde{\mathbf{s}}(\tau)$ has to be iteratively updated until convergence (i.e. the root mean square between $\tilde{\mathbf{s}}(\tau)$ and $\tilde{\mathbf{s}}(\tau - 1) < \epsilon$). The convergence rate depends on the eigenvalues of the normalized adjacency matrix which depends on the graph structure. The time complexity is linear in the number of steps τ needed to converge and linear in the number of edges $|E|$, since the product $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ requires to go through every entry (links) of the sparse matrix $\mathbf{A}$ just once. Finally, to apply the sum-of-similarities rule, we should compute $\tilde{\mathbf{s}}(\tau)$ for every class and then, as usual, assign the each node to the class with the maximum score.

6.1.3 The Approximate Sum-over-Paths Covariance

This first approach starts from the SoP covariance kernel. According to this measure, two nodes are considered as highly correlated if they often co-occur together on the same – preferably short – paths. This leads to the definition of a covariance kernel capturing similarities between pairs of nodes.

Let us first reintroduce the matrix $\mathbf{W}$, corresponding to Equation (3.8),

$$\mathbf{W} = \mathbf{P} \circ \exp\left[-\theta \mathbf{C}\right] = \exp\left[-\theta \mathbf{C} + \ln \mathbf{P}\right], \tag{6.4}$$

where $\mathbf{P}$ is the transition matrix containing the $p_{kk'}$, and the logarithm/exponential functions are taken elementwise. Moreover, $\circ$ is the elementwise (Hadamard) matrix product. If we pose $\mathbf{Z} = (\mathbf{I} - \mathbf{W})^{-1}$, then the **SoP covariance** between node k and node l (see Equation(4.37)) is

$$\begin{aligned} \mathrm{cov}(k,l) \;=\; & \frac{1}{\mathcal{Z}}\Big\{(z_{\bullet k}-1)z_{k\bullet}\delta_{kl} + z_{k\bullet}(z_{\bullet l}-1)(z_{lk}-\delta_{lk}) \\ & + z_{l\bullet}(z_{\bullet k}-1)(z_{kl}-\delta_{kl}) - \frac{z_{k\bullet}z_{l\bullet}(z_{\bullet k}-1)(z_{\bullet l}-1)}{\mathcal{Z}}\Big\} \end{aligned} \tag{6.5}$$

where $z_{k\bullet} = \sum_{k'=1}^{n} z_{kk'}$, $z_{\bullet k} = \sum_{k'=1}^{n} z_{k'k}$, $z_{\bullet\bullet} = \sum_{k,k'=1}^{n} z_{kk'}$, $z_{kk'}$ is element k,k' of $\mathbf{Z}$, $\mathcal{Z} = z_{\bullet\bullet} - n$, and δ_{kl} is the Kronecker delta whose value is 1 if $k = l$ and 0 otherwise. On the other hand, the **SoP betweenness centrality** measure (see Equation (4.36), is

$$\mathrm{bet}(k) = \frac{(z_{\bullet k}-1)z_{k\bullet}}{\mathcal{Z}} \tag{6.6}$$

and corresponds to the expected number of times node k appears on a path through the network.

As already mentioned, the goal here is to directly approximate $\mathbf{s}^c = \mathbf{K}\mathbf{y}^c$ where $\mathbf{K}$ is the **SoP covariance kernel matrix** containing the elements $\mathrm{cov}(k,l)$ (see Equation (6.5)). For the sake of readibility, let us fix a specific class c in the remainder of this Section 6.1, and omit the superscript c from $\mathbf{y}^c$. Now the sum of similarities between node k and the labeled nodes (of class c) is

$$\begin{aligned} & \sum_l \mathrm{cov}(k,l)\, y_l \\ = \;& \frac{1}{\mathcal{Z}}\left(z_{\bullet k}\sum_l z_{\bullet l}z_{lk}\, y_l + z_{\bullet k}\sum_l z_{l\bullet}z_{kl}\, y_l - z_{k\bullet}\sum_l z_{lk}\, y_l - \sum_l z_{l\bullet}z_{kl}\, y_l - z_{\bullet k}z_{k\bullet}y_k + z_{k\bullet}y_k\right) \\ & - \frac{1}{\mathcal{Z}^2}\left(z_{k\bullet}z_{\bullet k}\sum_l z_{l\bullet}z_{\bullet l}\, y_l - z_{k\bullet}z_{\bullet k}\sum_l z_{l\bullet}\, y_l - z_{k\bullet}\sum_l z_{l\bullet}z_{\bullet l}\, y_l + z_{k\bullet}\sum_l z_{l\bullet}\, y_l\right) \end{aligned} \tag{6.7}$$

If we denote element k of a vector $\mathbf{x}$ as $[\mathbf{x}]_k$ and the diagonal matrix constructed from

the vector $\mathbf{y}$ as $\mathbf{Diag}(\mathbf{y})$ and pose

$$\mathbf{x}_0 = \mathbf{Ze}, \text{ thus } z_{k\bullet} = \sum_l z_{kl} = [\mathbf{x}_0]_k \tag{6.8}$$

$$\mathbf{x}_1 = \mathbf{Z}^{\mathrm{T}}\mathbf{e}, \text{ thus } z_{\bullet k} = \sum_l z_{lk} = [\mathbf{x}_1]_k \tag{6.9}$$

$$\mathbf{x}_2 = \mathbf{Z}^{\mathrm{T}}\mathbf{Diag}(\mathbf{y})\,\mathbf{x}_1, \text{ thus } \sum_l z_{\bullet l} z_{lk}\, y_l = [\mathbf{x}_2]_k \tag{6.10}$$

$$\mathbf{x}_3 = \mathbf{Z}\,\mathbf{Diag}(\mathbf{y})\,\mathbf{x}_0, \text{ thus } \sum_l z_{l\bullet} z_{kl}\, y_l = [\mathbf{x}_3]_k \tag{6.11}$$

$$\mathbf{x}_4 = \mathbf{Z}^{\mathrm{T}}\mathbf{y}, \text{ thus } \sum_l z_{lk}\, y_l = [\mathbf{x}_4]_k \tag{6.12}$$

$$x_5 = \mathbf{x}_1^{\mathrm{T}}\,\mathbf{Diag}(\mathbf{y})\,\mathbf{x}_0 = \sum_l z_{l\bullet} z_{\bullet l}\, y_l \tag{6.13}$$

$$x_6 = \mathbf{e}^{\mathrm{T}}\,\mathbf{Diag}(\mathbf{y})\,\mathbf{x}_0 = \sum_l z_{l\bullet}\, y_l \tag{6.14}$$

then Equation (6.7) can be rewritten as

$$\begin{aligned}
&\sum_l \mathrm{cov}(k,l)\, y_l \\
&= \frac{1}{\mathcal{Z}}\Big([\mathbf{x}_0]_k[\mathbf{x}_2]_k + [\mathbf{x}_1]_k[\mathbf{x}_3]_k - [\mathbf{x}_0]_k[\mathbf{x}_4]_k - [\mathbf{x}_3]_k - [\mathbf{x}_1]_k[\mathbf{x}_0]_k[\mathbf{y}]_k + [\mathbf{x}_0]_k[\mathbf{y}]_k\Big) \\
&\quad - \frac{1}{\mathcal{Z}^2}\Big([\mathbf{x}_0]_k[\mathbf{x}_1]_k x_5 - [\mathbf{x}_0]_k[\mathbf{x}_1]_k x_6 - [\mathbf{x}_0]_k x_5 + [\mathbf{x}_0]_k x_6\Big)
\end{aligned} \tag{6.15}$$

where the partition function $\mathcal{Z}$ is computed by $z_{\bullet\bullet} - n = \mathbf{x}_0^{\mathrm{T}}\mathbf{e} - n$ (where $\mathbf{e}$ is a column vector full of 1's). The algorithm for computing this quantity consists in first solving the two systems of linear equations (6.8) and (6.9), which may be solved iteratively. For example, here is the way Equation (6.8) is solved

$$\begin{aligned}
\mathbf{Ze} = \mathbf{x}_0 \quad &\Longrightarrow \quad (\mathbf{I} - \mathbf{W})^{-1}\mathbf{e} = \mathbf{x}_0 \\
&\Longrightarrow \quad \mathbf{e} = (\mathbf{I} - \mathbf{W})\mathbf{x}_0 \\
&\Longrightarrow \quad \mathbf{x}_0 = \mathbf{W}\mathbf{x}_0 + \mathbf{e}
\end{aligned} \tag{6.16}$$

since $\rho(\mathbf{W}) < 1$, we may deduce the following iterative updating scheme

$$\begin{cases} \widehat{\mathbf{x}}_0(0) \leftarrow \mathbf{e} \\ \widehat{\mathbf{x}}_0(t+1) \leftarrow \mathbf{W}\widehat{\mathbf{x}}(t) + \mathbf{e} \end{cases} \tag{6.17}$$

Of course, more sophisticated methods, like conjugate gradient techniques or Gauss-seidel (see, e.g., Golub and Loan, 1996), could be used instead. Afterwards, using the same procedure, we solve the systems of linear equations (6.10), (6.11) and (6.12)

which are again solved iteratively. Finally, Equations (6.13) and (6.14) are computed directly. The complexity of each iteration is $\mathcal{O}(|E|)$ since the matrix is sparse. So the global complexity is approximately $\mathcal{O}(\tau|E||\mathcal{L}|)$ where τ is the number of iterations. Note that we might directly compute the results for all the $\mathbf{y}^c$ labeling vector classes by using (instead of a column vector $\mathbf{y}^c$) the concatenation of these vectors in a matrix $\mathbf{Y}$. In this case, depending on the architecture, solving the equations may be transparently parallelized. Indeed, we can easily compute the covariance for each class independently of the others. The computation may thus be parallelized directly on different cores, or CPUs. Finally, the assigned class index is provided by $\text{argmax}_c(\mathbf{K}\mathbf{y}^c)$ for all nodes.

Moreover, from Equation (4.36), the SoP betweenness centrality of node k can be computed from $\text{bet}(k) = ([\mathbf{x}_1]_k - 1)[\mathbf{x}_0]_k / \mathcal{Z}$.

6.1.4 The Bounded Normalized Random Walk with Restart

In this second approach, we will approximate a second graph-based method, referred to as the normalized random walk with restart (and called the regularized commute-time kernel in Fouss et al. (2007b)), by bounding the underlying random walk. The notion of approximating stationary distributions of random walks by only considering paths up to a specified length τ has been already investigated in (see, e.g., Callut et al., 2008; Sarkar and Moore, 2007; Rao et al., 2008), (for more details, see the related work in Section 6.4). We propose to apply this idea on the normalized random walk with restart kernel, and provide an interesting interpretation of the bounding. This kernel on a graph is closely related to the modified Laplacian matrix (see, e.g., Ito et al., 2005), the commute-time kernel (see, e.g., Saerens et al., 2004; Fouss et al., 2007a) and the well-known random walk with restart similarity (see, e.g., Pan et al., 2006; Tong et al., 2007). The normalized random walk with restart matrix $\mathbf{K}$ is given by

$$\mathbf{K} = (\mathbf{D} - \alpha\mathbf{A}^\mathsf{T})^{-1} \tag{6.18}$$

where $\mathbf{A}$ is the adjacency matrix, $\mathbf{D} = \mathbf{Diag}(\mathbf{A}\mathbf{e})$, and $\mathbf{e}$ is a column vector full of 1's. If matrix $\mathbf{A}$ is symmetric, Equation (6.18) defines a valid kernel on a graph. As shown now, the parameter $\alpha \in]0, 1[$ denotes, at each time step of a random walk, the probability that the random walker continues his walk. We are looking for a way to bound the sum-of-similarities $\mathbf{s} = \mathbf{K}\mathbf{y}$ up to a-priori-specified walk length τ. Following Fouss et al. (2007b), since the transition matrix of the natural random walk

on the graph is $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, we may note that

$$\begin{aligned}
\mathbf{K}\mathbf{y} &= (\mathbf{D} - \alpha\mathbf{A}^\top)^{-1}\mathbf{y} && (6.19)\\
&= \mathbf{D}^{-1}\mathbf{D}(\mathbf{D} - \alpha\mathbf{A}^\top)^{-1}\mathbf{y} && (6.20)\\
&= \mathbf{D}^{-1}((\mathbf{D} - \alpha\mathbf{A}^\top)\mathbf{D}^{-1})^{-1}\mathbf{y} && (6.21)\\
&= \mathbf{D}^{-1}(\mathbf{I} - \alpha\mathbf{P}^\top)^{-1}\mathbf{y} && (6.22)\\
&= \mathbf{D}^{-1}\left(\sum_{t=0}^{\infty}(\alpha\mathbf{P}^\top)^t\right)\mathbf{y} = \widehat{\mathbf{s}}(\infty) && (6.23)
\end{aligned}$$

Thus, intuitively, random walkers start from the labeled nodes with an initial distribution $\mathbf{x}(0) = \mathbf{y}$. Then, they diffuse with transition matrix $\alpha\mathbf{P}$, which is substochastic. Therefore, these random walkers have a non-zero probability of disappearing (giving up the walk) at each time step. Thus the random walkers are not restarting with a non-zero probability but rather disappearing. Let us denote by $\mathbf{x}(t)$ the column vector containing the expected number of random walkers in a specific node of the network after t steps of the random walk; thus, $\mathbf{x}(t) = \alpha\mathbf{P}^\top\mathbf{x}(t-1)$. Equation (6.23) tells us that the sum of similarities $\mathbf{K}\mathbf{y}$ is simply the normalized cumulated sum of expected visits to each node, $\mathbf{K}\mathbf{y} = \mathbf{D}^{-1}\sum_{t=0}^{\infty}\mathbf{x}(t)$. Bounding the walk aims to truncate this series up to term τ, $\widehat{\mathbf{s}}(\tau) = \mathbf{D}^{-1}\sum_{t=0}^{\tau}\mathbf{x}(t)$. The normalizing factor $\mathbf{D}^{-1}$ has the effect of normalizing the intrinsic importance of the nodes (see, e.g., Liben-Nowell and Kleinberg, 2007; Brand, 2005). Since $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, it can easily be shown that $\widehat{\mathbf{s}}(t)$ may be computed using the following recurrence equation:

$$\begin{cases}
\tilde{\mathbf{s}}(0) \leftarrow \mathbf{y} \\
\tilde{\mathbf{s}}(t) \leftarrow \alpha\,\mathbf{P}^\top\,\tilde{\mathbf{s}}(t-1) + \tilde{\mathbf{s}}(0), \quad \text{for } t = 1\ldots\tau \\
\widehat{\mathbf{s}}(\tau) \leftarrow \mathbf{D}^{-1}\tilde{\mathbf{s}}(\tau)
\end{cases} \tag{6.24}$$

This recurrence scheme can be iterated until convergence, but, in our experiments, we stop the iteration at $t = \tau$ steps, which is equivalent to bounding the random walk up to τ steps. The parameter τ will be tuned by cross-validation.

Notice that the matrix $(\mathbf{I} - \alpha\mathbf{P}^\top)^{-1}$ in Equation (6.22) coincides with the well-known random-walk with restart similarity matrix; it was used by Pan et al. (e.g. 2006) for computing similarities between nodes and was inspired from Page et al.'s famous PageRank algorithm. Equation (6.24) holds for directed graphs as well, but in this case the similarity matrix $\mathbf{K}$ is no more a valid kernel. The time complexity of this algorithm is $\mathcal{O}(\tau|E||\mathcal{L}|)$, which is the same as that of the SoP approximation presented in Section 6.1.3. Note, however, that we only have one system of linear equations to solve, while the SoP approximation requires solving five systems of the same size. In terms of spatial complexity, we only need to maintain one column vector at each time step for the current results and to store into memory the column vector $\mathbf{x}(0)$ and the

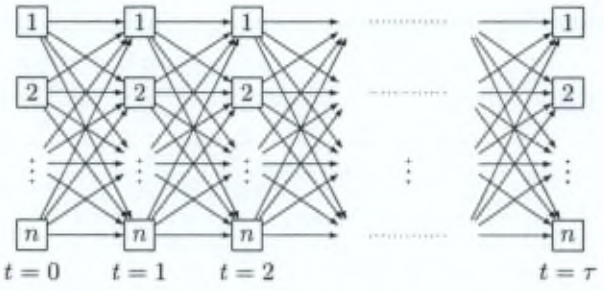


Figure 6.1: A lattice L defined from the original graph $\mathcal{G}$.

sparse transition matrix $\mathbf{P}$. $\mathbf{x}(0)$ needs to be computed only once at the initialization time. Therefore, the space complexity is $\mathcal{O}(|E| + |V|)$. Finally, the assigned class index is provided by $\text{argmax}_c(\mathbf{s}^c)$ for all nodes.

6.2 The Biased $\mathcal{D}$-walks

Callut et al. (2008) recently proposed still another random-walk based approach: the *discriminative random walks* ($\mathcal{D}$-walks), providing a class betweenness measure for classifying nodes in a graph. It computes a *group betweenness* index (see, e.g., Wasserman and Faust, 1994) with respect to a set of nodes – in this case, the labeled nodes belonging to the same class. This model performed well on a number of semi-supervised tasks (see Callut et al., 2008, for more details). In this section, we propose an extension of the $\mathcal{D}$-walks by using the randomized shortest path (RSP) framework introduced by Saerens et al. (2009). By defining an entropy-related parameter θ that controls the global entropy spread by the random walker in the network, we may gradually bias the random walk towards short paths. For a parameter value of $\theta = 0$, our extension reduces to the $\mathcal{D}$-walks. On the other hand, for intermediate values of θ, the random walk is biased towards short paths, therefore avoiding, to a certain extend, loops or broad, irrelevant, walks. (For additional motivations, the reader is advised to turn to Saerens et al., 2009).

A $\mathcal{D}$-walk relative to class c (see Callut et al., 2008, for further details) is a random walk on a graph that starts (at $t = 0$) from a node belonging to class c and ends in a node of the same class c. Therefore, the nodes of class c are transformed into absorbing nodes when $t \geq 1$. More precisely, the approach considered here consists in applying the randomized shortest path framework on a lattice structure constructed from the original network. Inspired by hidden Markov models (see, e.g., Rabiner, 1989), the main idea is the following: the original network $\mathcal{G}$ is *unfolded in time* in order to build a lattice L made of the network nodes at time steps $0, 1, \ldots, \tau$. Transitions are only allowed from nodes at time t to successor nodes at time $t + 1$ (see Figure 6.1). The

interpretation of this lattice is immediate: it represents a bounded random walk on the graph $\mathcal{G}$ with $t \in [0, \tau]$. The random walker starts in a node belonging to class c at time $t = 0$ and walks until either he reaches a node in class c, and is absorbed by this node, or stops at time $t = \tau$, that is, τ is the maximum walk length. The lattice L therefore contains $N = n \times (\tau + 1)$ nodes in total.

This lattice will be considered as a new graph – which is acyclic this time – on which the randomized shortest-path framework can be applied (see, e.g., Saerens et al., 2009). Its $N \times N$ exponential costs matrix is denoted by $\boldsymbol{W}^c$ and is organized in $(\tau+1)\times(\tau+1)$ blocks of size $n \times n$:

$$\boldsymbol{W}^c = \left(\begin{array}{c|c|c|c|c} \mathbf{0} & \widetilde{\mathbf{W}}^c(1) & \mathbf{0} & \dots & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \widetilde{\mathbf{W}}^c(2) & \dots & \mathbf{0} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \widetilde{\mathbf{W}}^c(\tau) \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{array}\right)$$

where $\mathbf{0}$ is a matrix full of zeroes of the appropriate size, i.e. $n \times n$ in this case.

Most blocks are null matrices since the graph considered here is a lattice and only transitions between time steps t and $t + 1$ are allowed. Hence, only submatrices at block $(t, t+1)$, (for $t = 0, 1, \dots, \tau$) may hold non-zero elements. Moreover, the matrices $\widetilde{\mathbf{W}}^c(t)$ are set equal to the $n \times n$ matrix $\mathbf{W}$ computed from Equation (3.8), with one important modification: when $t > 1$, the rows of the matrix corresponding to nodes labeled as class c are *replaced by zero rows*. This method aims at making these nodes absorbing: when a random walker hits one of these nodes, he stops his walk and disappears, following the strategy of $\mathcal{D}$-walks.

Two N-dimensional vectors, $\boldsymbol{h}_0^c$ and $\boldsymbol{h}_f^c$, are also defined,

$$\boldsymbol{h}_0^c = \left(\begin{array}{c} \widetilde{\mathbf{h}}^c \\ \hline \mathbf{0} \\ \hline \vdots \\ \hline \mathbf{0} \\ \hline \mathbf{0} \end{array}\right), \quad \boldsymbol{h}_f^c = \left(\begin{array}{c} \mathbf{0} \\ \hline \widetilde{\mathbf{h}}^c \\ \hline \vdots \\ \hline \widetilde{\mathbf{h}}^c \\ \hline \widetilde{\mathbf{h}}^c \end{array}\right)$$

These column vectors are indicator vectors defining the sets of starting and ending nodes of the random walks. The entries of the vector h_0^c (h_f^c) are set to 1 if a path can start from (or end into) the corresponding node, and 0 otherwise. They are also decomposed in $n \times 1$ blocks: $\widetilde{\mathbf{h}}^c$ is a $n \times 1$ vector whose values are equal to 1 if the node considered belongs to class c and 0 otherwise. In other words, $\widetilde{\mathbf{h}}^c = \mathbf{y}^c$. Since the paths considered start at time step 0, the vector h_0^c holds zero elements for all time steps $t > 0$. The vector h_f^c contains the $n \times 1$ vector $\widetilde{\mathbf{h}}^c$ for every time step larger than 0 since the random walker may stop at each time step $t = 1, 2, \ldots, \tau$ (except $t = 0$), when hitting a node of class c. For readability reasons, the superscript c of h_0^c and h_f^c is omitted in the remaining of this section.

The betweenness we are looking for is computed for each class c independently. Indeed, for each class, we consider all paths of length up to τ (in number of steps), starting from a node belonging to class c, and ending in a node of the same class. In other words, the betweenness measures to which extend a node is located in-between the nodes of the class of interest.

As detailed by Saerens et al. (2009); Yen et al. (2008) and already mentioned in Subection 6.1.3, an important quantity appearing in the RSP framework is $\boldsymbol{Z}^c = (\mathbf{I} - \boldsymbol{W}^c)^{-1}$, called the fundamental matrix. Every quantity of interest can be obtained from this matrix (see, e.g., Saerens et al., 2009, for details). For instance, the expected number of transitions through link $k \to k'$ (see Appendix A, Equation (A.7)) from which our betweenness will be derived, is

$$\overline{\eta}_{kk'}^c = -\frac{1}{\theta} \frac{\boldsymbol{h}_0^{\mathsf{T}} \boldsymbol{Z}^c (\boldsymbol{W}^c)'_{kk'} \boldsymbol{Z}^c \boldsymbol{h}_f}{\mathcal{Z}^c} \tag{6.25}$$

where $(\boldsymbol{W}^c)'_{kk'}$ is the partial derivative of $\boldsymbol{W}^c$ with respect to $c_{kk'}$ and $\mathcal{Z}^c = \boldsymbol{h}_0^{\mathsf{T}} \boldsymbol{Z}^c \boldsymbol{h}_f$. If we define $(\boldsymbol{\alpha}^c)^{\mathsf{T}} = \boldsymbol{h}_0^{\mathsf{T}} \boldsymbol{Z}^c$ as the forward parameters vector by reference to hidden Markov models (see, e.g., Rabiner, 1989). Since we only consider walks starting from class c and ending in class c, $\boldsymbol{h}_0^{\mathsf{T}}$ replaces $\boldsymbol{e}^{\mathsf{T}}$ and $\boldsymbol{h}_f$ replaces $\boldsymbol{e}$ in Equation (A.7).

Then,

$$\begin{aligned}
(\boldsymbol{\alpha}^c)^{\mathsf{T}} = \boldsymbol{h}_0^{\mathsf{T}} \boldsymbol{Z}^c \quad &\Longrightarrow \quad (\boldsymbol{\alpha}^c)^{\mathsf{T}} (\mathbf{I} - \boldsymbol{W}^c) = \boldsymbol{h}_0^{\mathsf{T}} && (6.26)\\
&\Longrightarrow \quad (\mathbf{I} - \boldsymbol{W}^c)^{\mathsf{T}} \boldsymbol{\alpha}^c = \boldsymbol{h}_0 && (6.27)\\
&\Longrightarrow \quad \boldsymbol{\alpha}^c = (\boldsymbol{W}^c)^{\mathsf{T}} \boldsymbol{\alpha}^c + \boldsymbol{h}_0 && (6.28)
\end{aligned}$$

Symmetrically, we define $\boldsymbol{\beta}^c = \boldsymbol{Z}^c \boldsymbol{h}_f$ as the backward parameters vector,

$$\begin{aligned}
\boldsymbol{\beta}^c = \boldsymbol{Z}^c \boldsymbol{h}_f \quad &\Longrightarrow \quad (\mathbf{I} - \boldsymbol{W}^c) \boldsymbol{\beta}^c = \boldsymbol{h}_f && (6.29)\\
&\Longrightarrow \quad \boldsymbol{\beta}^c = \boldsymbol{W}^c \boldsymbol{\beta}^c + \boldsymbol{h}_f && (6.30)
\end{aligned}$$

Decomposing $\boldsymbol{\alpha}^c$ in blocks in Equation (6.28) yields

$$\begin{pmatrix} \widetilde{\boldsymbol{\alpha}}^c(0) \\ \widetilde{\boldsymbol{\alpha}}^c(1) \\ \\ \\ \vdots \\ \\ \\ \widetilde{\boldsymbol{\alpha}}^c(\tau-1) \\ \widetilde{\boldsymbol{\alpha}}^c(\tau) \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ (\widetilde{\mathbf{W}}^c(1))^\intercal & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\widetilde{\mathbf{W}}^c(2))^\intercal & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & (\widetilde{\mathbf{W}}^c(\tau))^\intercal & \mathbf{0} \end{pmatrix} \begin{pmatrix} \widetilde{\boldsymbol{\alpha}}^c(0) \\ \widetilde{\boldsymbol{\alpha}}^c(1) \\ \\ \\ \vdots \\ \\ \\ \widetilde{\boldsymbol{\alpha}}^c(\tau-1) \\ \widetilde{\boldsymbol{\alpha}}^c(\tau) \end{pmatrix} + \begin{pmatrix} \widetilde{\mathbf{h}}^c \\ \mathbf{0} \\ \\ \\ \vdots \\ \\ \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

which allows to obtain the following forward recurrence relations:

$$\begin{cases} \widetilde{\boldsymbol{\alpha}}^c(0) & = \widetilde{\mathbf{h}}^c \\ \widetilde{\boldsymbol{\alpha}}^c(t+1) & = (\widetilde{\mathbf{W}}^c(t+1))^\intercal \, \widetilde{\boldsymbol{\alpha}}^c(t) \end{cases} \tag{6.31}$$

Similarly, we do the same for $\boldsymbol{\beta}^c$ to obtain the backward recurrence relations:

$$\begin{cases} \widetilde{\boldsymbol{\beta}}^c(\tau) & = \widetilde{\mathbf{h}}^c \\ \widetilde{\boldsymbol{\beta}}^c(t-1) & = \widetilde{\mathbf{W}}^c(t) \, \widetilde{\boldsymbol{\beta}}^c(t) + \widetilde{\mathbf{h}}^c, \; t-1>0 \end{cases} \tag{6.32}$$

Replacing $\boldsymbol{h}_0^T \boldsymbol{Z}^c$ (Equation (6.26)) and $\boldsymbol{Z}^c \boldsymbol{h}_f$ (Equation (6.29)) in Equation(6.25) yields

$$\overline{\eta}_{kk'}^c = -\frac{1}{\theta} \frac{(\boldsymbol{\alpha}^c)^\intercal (\boldsymbol{W}^c)'_{kk'} \boldsymbol{\beta}^c}{\mathcal{Z}^c} \tag{6.33}$$

The partition function $\mathcal{Z}^c$ corresponds to the contribution of all the paths starting and ending in class c. As stated, $\mathcal{Z}^c = \boldsymbol{h}_0^\intercal \boldsymbol{Z}^c \boldsymbol{h}_f$ and $\boldsymbol{\beta}^c = \boldsymbol{Z}^c \boldsymbol{h}_f$, hence, $\mathcal{Z}^c = \boldsymbol{h}_0^\intercal \boldsymbol{\beta}^c$. Since the $N \times 1$ column vector $\boldsymbol{h}_0$ is formed by a $n \times 1$ block which is $\widetilde{\mathbf{h}}^c$ and 0 values in the remaining positions, the product $\boldsymbol{h}_0^\intercal \boldsymbol{\beta}^c$ equals $(\widetilde{\mathbf{h}}^c)^\intercal \, \widetilde{\boldsymbol{\beta}}^c(0)$. Intuitively, it means that only paths starting at time step 0 contribute in the partition function.

$$\overline{\eta}_{kk'}^c = -\frac{1}{\theta} \frac{(\boldsymbol{\alpha}^c)^\intercal (\boldsymbol{W}^c)'_{kk'} \boldsymbol{\beta}^c}{(\widetilde{\mathbf{h}}^c)^\intercal \, \widetilde{\boldsymbol{\beta}}^c(0)} \tag{6.34}$$

Now, differentiating $\boldsymbol{W}^c$ with respect to $c_{kk'}$ and assuming that node k does not belong to class c – in which case we obtain the trivial result $\overline{\eta}_{kk'}^c = 0$ since k is absorbing –

yields

$$(\boldsymbol{W}^c)'_{kk'} = -\theta\, p_{kk'} \exp[-\theta c_{kk'}] \begin{pmatrix} 0 & \widetilde{\mathbf{e}}_k\widetilde{\mathbf{e}}_{k'}^\mathsf{T} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \widetilde{\mathbf{e}}_k\widetilde{\mathbf{e}}_{k'}^\mathsf{T} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \widetilde{\mathbf{e}}_k\widetilde{\mathbf{e}}_{k'}^\mathsf{T} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \widetilde{\mathbf{e}}_k\widetilde{\mathbf{e}}_{k'}^\mathsf{T} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

where $\widetilde{\mathbf{e}}_k$ is a $n \times 1$ column vector containing a 1 in position k, and 0's everywhere else. Therefore, the expected number of transitions through link $k \to k'$ is simply the sum of the expected number of transitions over all time steps,

$$\overline{\eta}^c_{kk'} = \frac{\sum_{t=0}^{\tau-1} (\widetilde{\boldsymbol{\alpha}}^c(t))^\mathsf{T} \widetilde{\mathbf{e}}_k \widetilde{\mathbf{e}}_{k'}^\mathsf{T} \widetilde{\boldsymbol{\beta}}^c(t+1)}{(\widetilde{\mathbf{h}}^c)^\mathsf{T} \widetilde{\boldsymbol{\beta}}^c(0)}\, w_{kk'} \tag{6.35}$$

with $w_{kk'} = \exp(-\theta c_{kk'} + \ln p_{kk'})$. Consequently, the betweenness of node k' with respect to class c is simply the sum of incoming transitions

$$\text{bet}^c_{k'} = \sum_{k=1}^{n} \overline{\eta}^c_{kk'} = \frac{\sum_{t=0}^{\tau-1} \sum_{k=1}^{n} \widetilde{\mathbf{e}}_k^T \widetilde{\boldsymbol{\alpha}}^c(t) \omega_{kk'} \widetilde{\boldsymbol{\beta}}^c(t+1) \widetilde{\mathbf{e}}_{k'}}{(\widetilde{\mathbf{h}}^c)^\mathsf{T} \widetilde{\boldsymbol{\beta}}^c(0)} \tag{6.36}$$

By observing from Equation (6.31) that $\sum_{k=1}^{n} \widetilde{\mathbf{e}}_k^\mathsf{T} \widetilde{\boldsymbol{\alpha}}^c(t) \omega_{kk'} = \sum_{k=1}^{n} \widetilde{\alpha}^c_k(t) \omega_{kk'} = \widetilde{\alpha}^c_{k'}(t+1)$, we finally obtain for the biased $\mathcal{D}$-walk betweenness vector

$$\mathbf{bet}^c = \frac{\sum_{t=0}^{\tau-1} \widetilde{\boldsymbol{\alpha}}^c(t+1) \circ \widetilde{\boldsymbol{\beta}}^c(t+1)}{(\widetilde{\mathbf{h}}^c)^\mathsf{T} \widetilde{\boldsymbol{\beta}}^c(0)} \tag{6.37}$$

where $\circ$ is the elementwise (Hadamard) matrix product. This betweenness is very similar to the $\gamma(t)$ variable computed in hidden Markov models (see, e.g., Rabiner, 1989; Jelinek, 1997) where it can be interpreted as the probability of being in some node after a walk of t steps without having visited any node of class c. Once the class betweenness has been computed, each node is assigned to the class showing the largest betweenness. The algorithm is detailed in Algorithm 2. The time complexity is $\mathcal{O}(\tau|E||\mathcal{L}|)$ since for each class $c \in |\mathcal{L}|$ we have to compute the forward and backward vectors which require τ steps each. Moreover, each vector computation imply a product between a sparse matrix and a column vector which is $\mathcal{O}(|E|)$. The space complexity is $\mathcal{O}(|E| + \tau|V|)$ since we have to store in memory both the matrix $\widetilde{\mathbf{W}}$ and the matrix

Algorithm 2 Computation of the biased $\mathcal{D}$-walk betweenness.

Input:

- A graph $\mathcal{G}$ containing n nodes.
- $\theta > 0$: the parameter controlling the degree of exploration.
- $\mathbf{C}$: the $n \times n$ cost matrix associated to $\mathcal{G}$, containing elements $c_{kk'} > 0$.
- $\mathbf{P}$: the $n \times n$ transitions-probabilities matrix of a natural random walk.
- $\widetilde{\mathbf{y}}^c$: the $n \times 1$ class membership vector containing 1 for nodes belonging to the class c and 0 otherwise.
- τ: the maximum walk length considered.
- $|\mathcal{L}|$: the number of classes.

Output:

- The **betweenness** matrix $\mathbf{B}$ containing $|\mathcal{L}|$ columns where each column contains the betweenness of each node relatively to a class.

1. $\widetilde{\mathbf{W}} = \mathbf{P} \circ \exp[-\theta \mathbf{C}]$, where the exponential is taken elementwise and $\circ$ is the elementwise (Hadamard) matrix product.
2. **for** $c = 1$ to $|\mathcal{L}|$ **do**
3. $\quad \widetilde{\mathbf{W}}^c = \widetilde{\mathbf{W}}$ and set the rows for which $\widetilde{\mathbf{y}}^c = 1$ to $\mathbf{0}^\mathsf{T}$.
4. $\quad \widetilde{\boldsymbol{\beta}}^c(\tau) = \widetilde{\mathbf{y}}^c$ (initialization of the backward vector).
5. $\quad$ **for** $t = \tau$ to 2 **do**
6. $\quad\quad \widetilde{\boldsymbol{\beta}}^c(t-1) = \widetilde{\mathbf{W}}^c \widetilde{\boldsymbol{\beta}}^c(t) + \widetilde{\mathbf{y}}^c$
7. $\quad$ **end for**
8. $\quad \widetilde{\boldsymbol{\alpha}}^c(1) = \widetilde{\mathbf{W}}^\mathsf{T} \widetilde{\mathbf{y}}^c$ (initialization of the forward vector).
9. $\quad$ **for** $t = 1$ to $\tau - 1$ **do**
10. $\quad\quad \widetilde{\boldsymbol{\alpha}}^c(t+1) = (\widetilde{\mathbf{W}}^c)^\mathsf{T} \widetilde{\boldsymbol{\alpha}}^c(t)$
11. $\quad$ **end for**
12. $\quad \mathbf{B}(:, c) = \frac{\sum_{t=1}^{\tau} \widetilde{\boldsymbol{\alpha}}^c(t) \circ \widetilde{\boldsymbol{\beta}}^c(t)}{(\widetilde{\mathbf{y}}^c)^\mathsf{T} \widetilde{\boldsymbol{\beta}}^c(0)}$ where $\circ$ is the elementwise multiplication.
13. **end for**
14. **return** $\mathbf{B}$

$\widetilde{\mathbf{W}}^c$ and at the same time we have to keep in memory τ times the forward and backward vectors.

6.3 Experiments

This experimental section has two main goals. Firstly, the three approximate algorithms introduced in this chapter are compared to their exact counterpart (without approximating or bounding) and to some state-of-the art techniques on several graph-based semi-supervised classification tasks over medium-size data sets. Secondly, the performance of our three proposed algorithms are further assessed and compared to two state-of-the art techniques on (i) a large 6-classes sparse network, consisting in the graph of citations between about 3 million U.S. patents and (ii) a standard large-scale secondary structure data set.

6.3.1 First Experiment: Validation of the Approximate Approaches

In this first part of the experiments, we address the task of classifying unlabeled nodes in partially labeled graphs on eight medium-size data sets (up to 5000 nodes). The goal is to compare the approximate approaches to the exact kernel-based techniques in terms of classification rate. This comparison is performed on medium-size networks only since kernel approaches cannot be computed on large networks.

Data sets. The different classification models, referred to as classifiers, are compared on eight data sets that were used previously for semi-supervised classification: the four universities `WebKB cocite` data sets (see, e.g., Zhou et al., 2005; Macskassy and Provost, 2007), the two `industry` data sets (see, e.g., Macskassy and Provost, 2007), the `IMDb prodco` data set (see, e.g., Macskassy and Provost, 2007) and the `CoRA cite` data set (see, e.g., Macskassy and Provost, 2007)*. These data sets are fully described in Section 5.3.

Classification methods. The standard kernel-based classifiers compared in this experiment are based on (1) the SoP covariance (SoP) kernel introduced in this thesis (see Subsection 6.1.3, Equation (6.5)), (2) the normalized random walk with restart (NRWR) kernel , a normalized version of the random walk with restart (see Subsection 6.1.4, Equation (6.18)). The approximate classification methods proposed in this work are (3) the approximate sum-over-paths classifier (aSoP, see Subsection 6.1.3), based on the SoP covariance kernel, (4) the bounded normalized random walk with restart classifier (bNRWR, see Subsection 6.1.4) and (5) the biased $\mathcal{D}$-walk (b$\mathcal{D}$WALK, see Section 6.2). Moreover, as a baseline, we report the results obtained by using (6) the normalized, regularized, Laplacian (NRL) kernel, $(\mathbf{I}+\alpha\widetilde{\mathbf{L}})^{-1}$, where $\widetilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}}$ is the normalized Laplacian matrix (see, e.g., Zhou et al., 2005) that achieved a competitive performance in the previous chapter, (7) the approximate normalized, regularized, Laplacian (aNRL) kernel which is the approximate counterpart of the NRL directly computing sum of similarities by solving iteratively (up to τ steps) the system of linear equation, (8) the hitting time (hit Time) $h(k|\{i \in c\})$ (also called the average first-passage time) which in this application measures the average time (i.e. average steps) a random walker starting from any node i of a specified class c will take to reach a unlabeled node k for the first time (see, e.g., Sarkar and Moore, 2007). The nodes hitting time for each possible diffusion class has been approximated by solving iteratively a system of linear equations until convergence. Afterwards, we assign each node k to the class for which its hitting time is a minimum ($\text{argmin}_{c\in\mathcal{C}}\ h(k|\{i \in c\})$). Finally, the results of a simple (9) k-nearest-neighbor (KNN) are also reported. Our implementation of the KNN consists in taking all neighbors at maximum k steps of the considered node. An unlabeled node will be labeled with the tag that is most represented in the

*The preprocessed version in Matlab/Octave format of the data sets used in this first experiment is available from `http://iridia.ulb.ac.be/~amantrac/pub/SoP_TPAMI.zip`

set of nodes located at maximum k steps (where duplicates have been removed). Each vote is weighted by the similarity in terms of number of steps ($1/k$) with the node of interest. Notice that for all the bounded methods (bNRWR, b$\mathcal{D}$WALK, KNN), the walk length τ is tuned during cross-validation. On the other hand, the approximate methods (aSoP, aNRL, hit Time) are iterated until convergence (RMSE $< 1.0\epsilon - 04$).

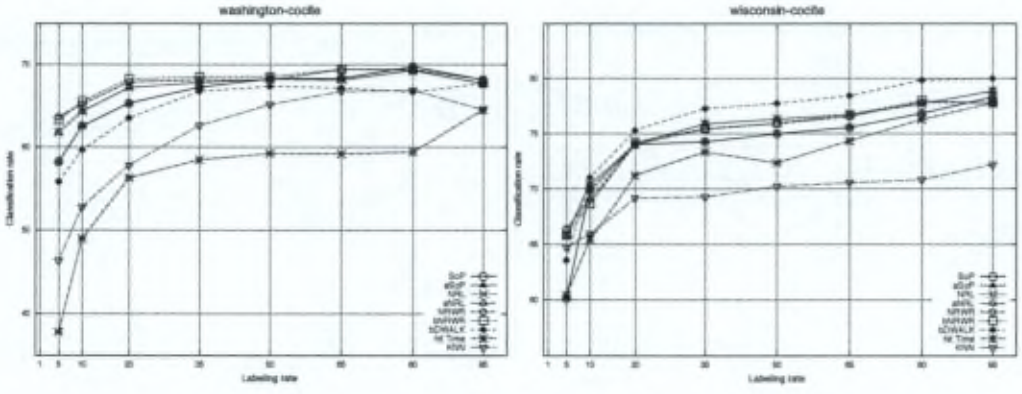
Remember that for all the kernel-based methods, the class label is obtained by computing the sum of similarities between the node of interest and the labeled nodes, as detailed in Subsection 6.1.1.

Experimental methodology. The classification accuracy will be reported for several labeling rates (5, 10, 20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The labels of remaining nodes are used as test data. For each considered labeling rate, 10 random node label deletions (test sets) were performed (10 runs), on which performances are averaged. For each unlabeled node, the various classifiers predict the most suitable category according to the procedures described previously. During each run, a 10-fold nested cross-validation is performed. The external folds are obtained by 10 successive rotations of the nodes and the performance of one run is averaged over these 10 folds. For each fold of the external cross-validation, a 5-fold internal cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (i.e., the parameter θ for SoP, aSoP and bDWALK, the parameters α for NRWR, bNRWR, NRL and aNRL, the walk length τ for bDWALK, bNRWR and aNRL, the parameter k of KNN). For each method and each labeling rate, we report the average classification rate averraged on the 10 runs.

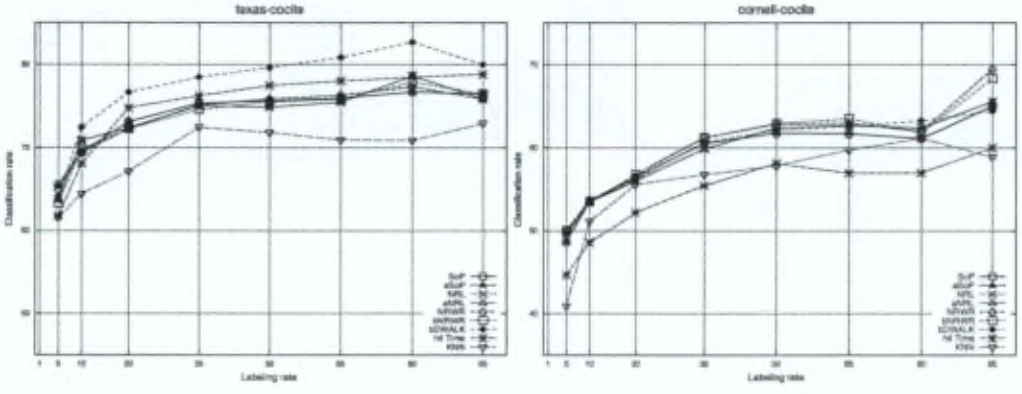
Results and discussion. Comparative results for each method on the eight different data sets are reported in Figures 6.2(a), 6.2(b), 6.3(a) and 6.3(b). Clearly, whatever the labeling rate considered, the approximate methods achieve almost the same performance as their exact kernel-based counterpart on all data sets (i.e. there is no significative difference in terms of accuracy according to a sign test for a labeling rate of 10% with a p-value of 0.01). By analyzing statistically the significance of the results (Table 6.1), we observe that the best methods overall are the NRWR (and it's bounded counterpart), the biased $\mathcal{D}$-walk, and the NRL (and it's approximated counterpart) since they range among the top methods on all the benchmarked data sets. The SoP approach obtains good results in general, but it achieves sometimes least competitive results for a low labeling rate. This is the case on the two industries data set (Figure 6.3(a)) and on the washington-cocite data set (Figure 6.2(a)). Finally the hitting time approach results seems to be least competitive on these data sets except for the texas-cocite.

6.3.2 Second Experiment: Application to Large-Scale Networks

In this second part of the experiments, we address the same task of classification of unlabeled nodes in partially labeled graphs, but this time on two large-scale data sets.



(a)



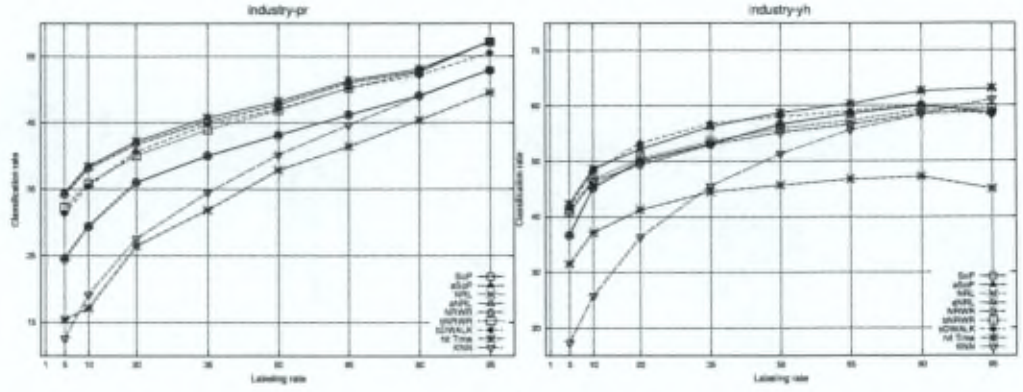
(b)

Figure 6.2: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, aSoP, NRWR, bNRWR, NRL, aNRL, bDWALK, hit Time and KNN classification methods. The graphs show the results obtained on the `washington`, `wisconsin`, `texas` and `cornell` `WebKB` data sets.**

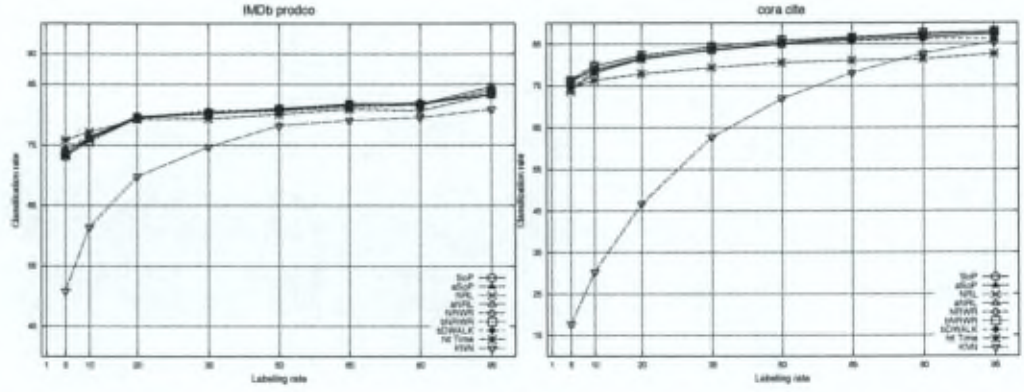
The goal is to compare the results obtained by the three new algorithms introduced in this chapter. The first data set considered is the `U.S. patents` network which consists of more than 3 millions of nodes interconnected by 38 millions of links. The second data set consists in an amino acid sequence window associated to a target secondary structure. This data set has already been used to investigate how far state-of-the-art semi-supervised methods can cope with large-scale application by Chapelle et al. (2006).

Data sets.

The patent data set introduced in this work is based on two publicly available databases: the NBER data set (see, e.g., Hall, 2001; Hall et al., 2005) and the PatStat database (see,



(a)



(b)

Figure 6.3: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, aSoP, NRWR, bNRWR, NRL, aNRL, b$\mathcal{D}$WALK, hit Time and KNN classification methods. The graphs show the results obtained on the two `industries` data sets and on the `IMDb` and `CoRA` data sets (see, e.g., Macskassy and Provost, 2007).**

e.g., Office, 2007). The resulting set is made of 3,416,966 U.S. patents granted between 1963 and 2002 and contains bibliographic data on each patent such as filing and grant dates, priority numbers (in case a U.S. patent was filed following preceding national or international applications), number of claims, and the list of countries of extension (countries other than the U.S. where the same patent was filed). In the economic literature, these data are intensively used to measure innovation, the output of research and development activities, or to estimate the value of patents.

In addition, the data set includes the names and residence address of all inventors and assignees (i.e. companies) listed on each patent. These data can be used to analyze the geographical origin of patents and inventions. Companies listed as assignees can be

Algorithm	aSoP	bNRWR	b$\mathcal{D}$WALK	hit Time	aNRL
aSoP	—	=(7), < (1)	= (8)	> (5), = (3)	= (7), < (1)
bNRWR	—	—	> (1), = (6), < (1)	> (7), = (1)	= (7), < (1)
b$\mathcal{D}$WALK	—	—	—	> (6), = (2)	= (7), < (1)
hit Time	—	—	—	—	= (1), < (7)

Table 6.1: **Compilation of statistical sign tests computed for the aSoP, bNRWR, b$\mathcal{D}$WALK and hit Time classification methods. A signed test have been performed on each of the 8 medium-size data sets, based on the results of the 10 runs of the semi-supervised classification task, for a fixed labeling rate of 10%. Each entry of the table shows the number of times the row method is respectively significantly (i.e. p-value < 0.01) better (>), the same (=) or worse (<) than the column method. For instance, bNRWR performs significantly better than hit Time on 7 data sets.**

matched with additional economic data such as turnover, profits, or stock value for the sake of economic analysis. The corpus is complemented with textual data including the English title and abstract of each patent. This information could be used in further work to label the patents based on both citations and text.

More importantly, patents are classified according to different U.S. and international classifications. The main U.S. class used in this work contains 6 broad industrial areas (chemicals, ICT, drugs and medical, electrical and electronic, mechanical, others), each of which contains up to 9 subclasses. On the other hand, the international patent classification (IPC), maintained by the World Intellectual Property Organization (WIPO), provides a hierarchical representation of all technological fields. The first level contains 8 classes (labeled from A to H: human necessities, transporting, chemistry and metallurgy, textiles and paper, fixed constructions, mechanical engineering, physics, electricity). The second level contains up to 20 subclasses (from 01 to 20). At the third level, the IPC class is made of 4 digits (from A01A to H10G). Additional levels are available up to 11 digits. It is to be noted however that patents are frequently assigned to several classes. In this case, one class is referred to as the main category. Although it is difficult to link such technological classes to industrial sectors, the economic literature has used them intensively to analyze innovation activities across industries (see, e.g., van Zeebroeck et al., 2006).

The graph structure in the data is made of *bibliographic references* between patents. In order to obtain a patent, an inventor must provide the list of references to patents and scholarly publications upon which the invention is based. This list is then completed by an examiner at the patent office in an attempt to delineate as clearly as possible the territory covered by each patent. What had been published earlier can indeed no longer be patented. The network of citations between patents can be seen as indicative of knowledge flows between companies, countries or industries. They have therefore been intensively used as indicators of knowledge spillovers in the economic literature (see, e.g., Jaffe et al., 1993; Cowan and Foray, 1997; Lichtenberg and van Pottelsberg

Category	Size	Proportion
Chemicals	630107	19,42%
ICT	381537	11,76%
Drugs and medical	245595	7,57 %
Electrical and electronic	575369	17,73 %
Mechanical	724022	22,31 %
Others	688375	21,21 %
Total	3245005	100%
Majority class proportion	22,31%	

Table 6.2: **Class distribution for the `U.S. patents` data set.**

Category	Size	Proportion
α-helical and β-sheet	35823	42,81%
coil	47856	57,19%
Total	83679	100%
Majority class proportion	57,19%	

Table 6.3: **Class distribution for the `Secondary Structure` data set.**

de la Potterie, 1996; Cowan and Foray, 2000; Cassiman and Veugelers, 2002). In addition, since patent citations indicate downstream research activities, hence investments around the same technology, they suggest that an intensively cited patent denotes a particularly important or valuable invention. Citation counts have therefore also been used to produce value-weighted counts of patents (see, e.g., Trajtenberg, 1990; Hall, 2001; Hall et al., 2005). One can naturally expect patents to preferably cite patents from the same technological class.

However, not all patents have citations to or from other patents, and many citations actually refer to pre-1963 patents for which most data are lacking. Excluding unconnected patents and records with missing values, the graph is left with 3,245,005 interconnected nodes and a total of 19,423,243 links.

The graph of patent citations is directed by nature, since patents can only cite earlier publications. However, for the sake of the algorithms implemented in this chapter, the matrix of links has been made symmetrical (as if each citing-cited pair of patents was a set of mutually citing patents). This matrix, in addition to the main class assigned to each patent in each classification scheme, is provided along with this thesis[†]. Note that official patent numbers have been replaced by sequential numbers (from 1 to 3,245,005) to ease computations. The class distribution of nodes in the `U.S. patents` network is shown in Table 6.2.

Furthermore, the SoP betweenness has been computed according to Equation (4.36). The histogram of the base-10 logarithm of this betweenness is displayed in Figure 6.4, for a θ parameter equal to 0.1 according to previous tests (see previous chapter).

[†]This preprocessed data set in Matlab/Octave format is available from `http://iridia.ulb.ac.be/~amantrac/pub/patents.mat.tar.gz`

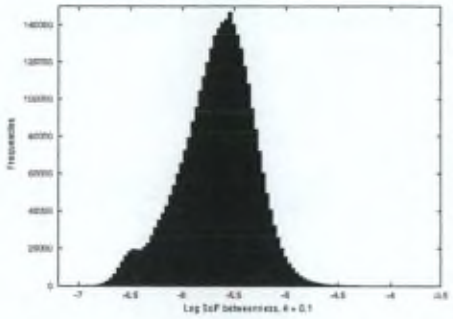


Figure 6.4: **Distribution of the logarithm of the SoP betweenness computed on the `U.S. patents` network. The θ parameter was fixed to** 0.1.

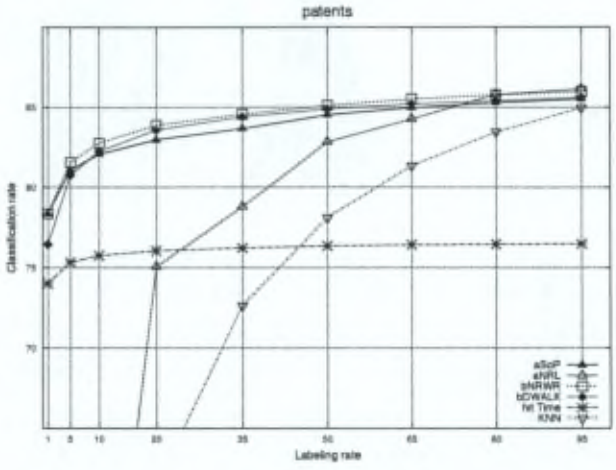
The betweenness scores are located in the interval $[7.3172e-08, 2.7947e-04]$. For information, the patents that obtain the highest betweenness scores are, respectively, the U.S. patent 4340581 on *DSCG binding protein and process for preparing same* of July 20, 1982; the U.S. patent 4683202 on *Process for amplifying nucleic acid sequences* of July 28, 1987 and the U.S. patent 4723129 on *Bubble jet recording method and apparatus in which a heating element generates bubbles in a liquid flow path to project droplets* of February 2, 1988.

The secondary structure (SecStr) is a large data set that has been benchmarked in Chapelle et al. (2006). It consists of 83679 amino acids around which an amino acids window$[-7, +7]$ is centered. The target is composed of two main classes: the α-helical and β-sheet secondary structure form one class of 47856 protein positions while the remaining coil structural motif positions form the other class. Hence, the main task is to predict the secondary structure of a given amino acid in a protein based on a sequence window. The class distribution of this data set is shown in Table 6.3.

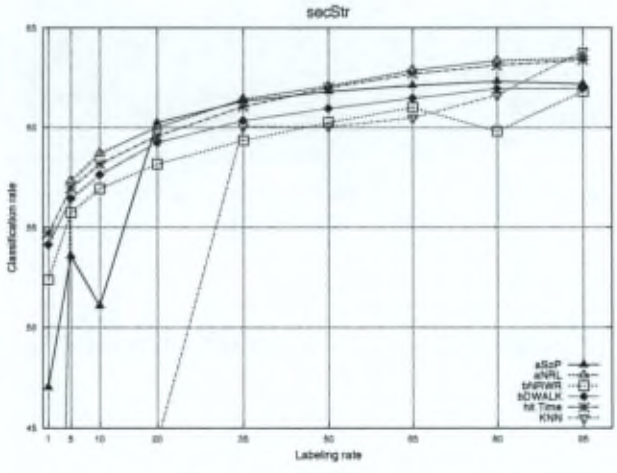
Classification models. The tested classifiers are (1) the SoP approximation (aSoP), (2) the bounded normalized random walk with restart kernel (bNRWR) (3) the bounded $\mathcal{D}$-walk (b$\mathcal{D}$WALK) and (4) the hitting time (hit Time). As baseline, the results of (5) a simple k nearest neighbor (KNN), as well as (6) the approximate normalized regularized Laplacian (aNRL), are also reported. In this experiment, the number of maximum considered steps for the KNN was limited to $k = 2$ because of computational issues.

Experimental methodology. The classification accuracy will be reported for increasing labeling rates (1, 5, 10, 20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The labels of remaining nodes are used as test data. For each considered labeling rate, 10 random node label deletions (test sets) were performed (10 runs), on which performances are averaged. For each run, a 10-fold cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (see the first experiment in Section 6.3.1 for details). Thus, the performance

on each run is assessed on the remaining unlabeled nodes with the hyper-parameter tuned during the 10-fold cross-validation.



(a)



(b)

Figure 6.5: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 1, 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the aSoP (approximate sum-over-paths), the bNRWR (bounded normalized random walk with restart), the b$\mathcal{D}$WALK (biased $\mathcal{D}$-walk), the aNRL (approximate normalized regularized Laplacian), the hit Time and the KNN methods. The graphs show the results obtained on the `patents` data set and the `secStr` data set**

Results and discussion. The results for each method and each labeling rate are reported on Figure 6.5(a) for the us patents and on Figure 6.5(b) for the amino acids sequence data set.

Size	aSoP	aNRL	bNRWR	b$\mathcal{D}$WALK	hit Time	KNN
medium-size	15.90	14.43	**14.22**	**15.09**	17.44	28.09
large-scale	11.17	68.66	**8.53**	**8.49**	5.50	64.30

Table 6.4: **Averaged accuracy drop obtained when labeling rate decreases from 95% to 5% on medium-size networks, and from 95% to 1% on large-scale networks. The accuracy, averaged over 10 runs, were obtained for medium-size networks on: `washington`, `wisconsin`, `texas`, `cornell` `WebKB`, `industries`, `IMDb`, `CoRA` data sets, and for large-scale network on: `U.S. patents citation network` and `secStr` data set. The results are reported for the aSoP, the bNRWR, the b$\mathcal{D}$WALK, the aNRL, the hit Time and the KNN.**

Algorithm	1%	5%	10%	20%	35%	50%	65%	80%	95%
aSoP	769	749	972	883	658	291	313	351	337
aNRL	45	15	17	25	46	77	134	179	246
bNRWR	41	42	31	82	118	178	261	380	505
b$\mathcal{D}$WALK	55	58	63	79	120	184	271	379	511

Table 6.5: **Overview of cpu time in seconds needed for running an algorithm (and thus classifying all the unlabeled nodes), averaged over 10 runs, obtained on the `U.S. patents network` for labeling rates of 1, 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the aSoP, the bNRWR, the b$\mathcal{D}$WALK and the aNRL. The cpu used is an Intel(R) Xeon(R) CPU E5335 @2.00GHz, with 4096 KB of cache size and 8GB of RAM.**

For the us patents (Figure 6.5(a)), we observe that the results are very stable across the 10 runs. The bounded normalized random walk with restart kernel-based method (bNRWR) achieves the best performance (significant, $p < 0.01$, at labeling rate of 5% and 10%, according to a t-test performed on the 10 runs) for all labeling rates, very closely followed by the b$\mathcal{D}$WALK (except for a very low labeling rate, where the b$\mathcal{D}$WALK performance drops), and closely followed by the aSoP. It can be observed that these three proposed techniques are not very sensitive to the labeling rate. Indeed, the drop in performance from 95% to 5% labeling rate is lower than 6%, no matter the method considered. The hitting time approach is also stable but it underperforms in comparison with the three top leading methods. Actually, we already observed on medium size data sets that the hitting time performance was somewhat below the top methods (see experiments on medium size data set Section 6.3.1). In contrast, we observe a significant drop in performance for the aNRL and the KNN when the labeling rate decreases. For the sake of readability the results of the KNN and the aNRL are not shown for low labeling rates. Actually, both methods obtain a classification rate around 30% and around 8% for respectively a labeling rate of 5% and 1%.

On the SecStr data set (Figure 6.5(a)), we observe a suddenly large drop of the aNRL curve below 5% of labeling rate while the drop is more gentle on the us patents network. The hitting time may be considered as the leading method on this data set, very closely

followed by the b$\mathcal{DW}ALK$ and the bNRWR — all these three algorithms are rather robust with respect to changes in labeling rate (i.e. the drop in performance is lower than 10%). Actually, only the bNRWR and the biased $\mathcal{DW}ALK$ achieved also very stable results on the us patents graph as well as and on the eight medium-size networks. Thus, according to our benchmarks, these two approaches seem to be quite robust. Finally, the aSoP results are a bit disappointing on this data set since its performance curve drops more sharply than the leading algorithms for low labeling rates. As before, for the sake of readability the results of the KNN and the aNRL are not reported for low labeling rates.

An analysis of the stability of the algorithms on all benchmarked data sets (medium-size and large-scale) is provided in Table 6.4. It shows, for all the implemented methods, the averaged accuracy drop when the labeling rate decreases from 95% to 5% on medium-size graphs, and from 95% to 1% on large-scale networks. The score obtained by the two most robust methods (i.e. bNRWR and the b$\mathcal{DW}ALK$) are indicated in bold.

The Table 6.5 provides a comparison of the running time of all methods, averaged over 10 runs for the `U.S. patents` network. We observe that the classification task only takes a few minutes, whatever the method used. The aSoP method is significantly slower, but is still able to classify the whole graph in a few minutes.

6.4 Related work

Graph-based semi-supervised classification has been the subject of intensive research in recent years. A wide range of approaches have been developed in order to address the problem. Among them, we may cite random walks (see, e.g., Zhou and Scholkopf, 2004; Szummer and Jaakkola, 2001; Callut et al., 2008), graph mincuts (see, e.g., Blum A., 2001), spectral methods (see, e.g., Chapelle et al., 2002; Smola and Kondor, 2003; Kondor and Lafferty, 2002; Kapoor et al., 2005), regularization frameworks (see, e.g., Belkin et al., 2004a; Wang et al., 2008; Yajima and Kuo, 2006; Zhou et al., 2003, 2005), transductive and spectral SVMs (see, e.g., Joachims, 2003). For a comprehensive survey of semisupervised classification, including graph-based approaches, see (see, e.g., Zhu, 2008; Zhu and Goldberg, 2009). Some of these approaches tackle the problem by random-walk based methods. However, it has been shown that hitting times and commute times approaches suffer from several problems; for example, they take too long paths (hence irrelevant) into account so that popular entities are intrinsically favored (see, e.g., Liben-Nowell and Kleinberg, 2007; Brand, 2005). Moreover, random-walk based techniques often require to inverse a matrix in order to compute measures on walks that are potentially of infinite length.

These shortcomings led some authors such as Sarkar et al. (2008); Sarkar and Moore (2007) or Callut et al. (2008) to consider bounded (or truncated) walks. Sarkar et al.

(2008); Sarkar and Moore (2007), for instance, used a truncated commute-time approach and showed experimentally that the truncation boosts the results while providing a competitive computing timein a proximity search task on a large graph with 600K nodes. In the same spirit, Callut et al. suggested to bound walks for tackling graph-based semi-supervised problems (see, e.g., Callut et al., 2008). Their approach offered a time complexity $\mathcal{O}(\tau|\mathcal{L}||E|)$, but no experimental results on large graphs were presented. In this chapter, we propose precisely a generalization of the algorithm introduced by Callut et al. by using the randomized shortest path framework (see, e.g., Saerens et al., 2009; Yen et al., 2008). In addition, we present experimental results on the large-scale `U.S. patents` data set.

Tong et al. (2006) suggested a method avoiding to inverse the complete matrix for computing the random walk with restart measure. Their idea is to reduce the computing time by partitioning the input graph into smaller communities. Then, by applying a low rank approximation, they were able to approximate the random walk with restart. The approximated matrix obtained is sparse and hence can be kept into memory. Furthermore, they reduce the computing time by the number of communities initially found. Thanks to the Sherman-Morrison lemma, the precomputed inverse is updated on-the-fly for a new query. This method suffers from the fact that it adds a hyperparameter k, e.g. the number of communities, that depends on the network. Furthermore, the computing time is reduced by the factor k which is still untractable for large graphs with millions of nodes.

Moreover, Tong et al. (2008, 2007) recently developed a direction-aware proximity method based on the concept of escape probability. They presented two methods to compute efficiently this proximity measure. The first one, FastAllDAP, allows to compute directly the proposed measure between all pairs of nodes by reducing the cost of solving a large number of linear systems to one matrix inversion. This method can only be applied on medium-size graphs (i.e. < 10K nodes). The second method, FastOneDAP, may be applied to large graphs, but only computes one measure. In this case, the matrix inversion problem is approximated by a Taylor expansion of the concerned matrix and the complexity is reduced to $\mathcal{O}(\tau|E|)$. In this work, we avoid the direct computation of the proximity measure by taking advantage of the property of our targeted task, i.e. semi-supervised classification. Using the FastOneDAP directly for this task would require computing the measure $|V|$ times (i.e., for each node), so that its time complexity would be $\mathcal{O}(\tau|E||V|)$. Recall that the time complexity of our proposed method is $\mathcal{O}(\tau|E||\mathcal{L}|)$. Since the number $|\mathcal{L}|$ of classes is usually much smaller than the number $|V|$ of nodes, we save an important amount of computing time.

Finally, Herbster et al. (see, e.g., Herbster et al., 2008) proposed a technique for fast label prediction on a generic graph through the approximation of the graph with either a minimum spanning tree or a shortest path tree. Once the tree has been extracted, they are able to compute the pseudoinverse of the Laplacian matrix – a well-known

kernel on a graph (see articles by Fouss et al., 2007a; Saerens et al., 2004; Hirai et al., 2005, among others) – efficiently. The fast computation of the pseudoinverse enables to address prediction problems on large graphs.

6.5 Conclusion

This work investigated several approaches for tackling semi-supervised problems as well as betweenness computation on large, sparse, graphs. While this chapter focuses on semi-supervised classification and betweenness computation, the same approaches (bounding or approximating random walks) could easily be applied in order to compute other graph measures, such as group degree centrality, closeness centrality, etc Wasserman and Faust (1994). This will undoubtedly be the subject of further research. We will also, as future work, pursue the analysis of the `U.S. patents` data set introduced in this chapter. For instance, the correlation between various measures of importance of the nodes (such as centrality/prestige) and econometric indicators of the value of a patent will be investigated. We are also studying the impact of using additional patent information such as the abstract, various econometric indexes, etc, on the results of semi-supervised classification. Another interesting issue in this respect is how to combine the information provided by the graph and the node features in an clever, preferably optimal, way. This last issue will be the subject of next Chapter.
Still another interesting application of the techniques presented in this chapter is collaborative recommendation. Indeed, the same methods could be used for large-scale recommendation (like the Netflix challenge) using graph kernels in the same way as Fouss et al. (2007a).

Chapter 7

Combining Graph Topology and Node Features

Graph-based semi-supervised learning techniques exploit graph topology for building a predictive model. Indeed, by exploiting the graph topology, labels are assigned to unlabeled nodes. As already seen in Chapter 5, by defining a regularization framework we are able to support global and local consistency assumptions. However, in real world applications, graphs constitute just one kind of information about the underlying data. While graphs capture the information of dependence between the data, each individual observation may also have a set of descriptive variables (i.e. features). Traditional data mining and statistical approaches are generally based on a table view model of the underlying data. Such models make the hypothesis that the data are independently and identically distributed (*i.i.d.*). However, it often happens that such an assumption is false since the observations may be interconnected. This is generally the case when the information forms a network of great interdependence between the different actors involved. Despite that, modeling such a network by a table view of independent observations is often required by state-of-the-art statistical techniques. On the other hand, novel graph-based techniques that are able to make use of the network structure linking the data constitute an important challenge. However, as we have seen so far, graph-based techniques generally do not exploit the feature information on nodes. Indeed, they usually exclusively use the topological information linking the different actors.

An interesting real world networks is indisputably the `U.S. patents` network (as seen in the previous chapter Section 6.3.2). While patents are interconnected through "cited, citing" relations, each of them also contains: a textual description, information about the owner, information about the company to which the patent belongs, the date of submission, etc. All these sources of information may be useful for improving the performance of graph-based models. Another example occurs in telecommunications

where operators are confronted with the churning problem: It concerns the decision taken by subscribers to change operator. Detecting such decisions is crucial in order to prevent escaping customers as long as possible. Such detection should certainly make use of the telecom network structure as well as the available descriptive information on customers habits. Another famous example is web page ranking, where both a graph structure and the page contents are available (see e.g. page-rank, hits, ...).
The goal of this chapter is to investigate different ways to reconcile these two visions of the world. The table view model casting aside the inner dependency of the data and the graph view model exploiting exclusively the structure of the graph.

Contributions and Organization of this Chapter

This chapter has three main contributions:

- It shows empirically, through systematic experiments on seven selected data sets, that citation graphs are more suited (i.e. obtain better classification rate) for classification than inferred content graphs.
- It shows empirically, through systematic experiments on seven selected data sets, that connecting documents to an external citation graph resource can significantly improve the accuracy for a semi-supervised classification task (i.e. we observe 10 to 30% increase of the classification rate depending on the data set).
- Finally, it shows, that combining a citation graph to the mined counterpart document similarity graph does not increase significantly the accuracy results, at least on seven selected data sets.

The subsequent part, Section 7.1, introduces different ways to infer a sparse content graph from a set of documents. In Section 7.2, graph-based semi-supervised methods working on an inferred content graph are compared to a state-of-the-art text categorization method (linear SVM) working on a bag of words representation of documents. Section 7.3 reinvestigates seven different data sets already encountered in the previous chapter. Content graphs are inferred from these data sets and different graph-based semi-supervised algorithms are applied on them. Then, we compare these results to those obtained on the citation counterpart graphs tested in the previous chapter. Section 7.4 proposes two different experiments by combining citation graphs to their mined content graphs. On the first experiment, we assess how well the classification rate of *citation nodes* can be improved. On the other hand, on the second experiment, we evaluate how well the classification rate of *document nodes* can be improved. In Section 7.5, a brief survey of the field is provided. Finally, the last part of the chapter includes conclusions and remarks as well as further extensions.

7.1 Graph Extraction from Text Documents

There are many ways for combining the topological information of a graph with information on its nodes (see, e.g., Macskassy, 2007). For example, the graph can be represented as a table, where each row stands for a node and each column a link to another node. This sparse column space, defined by the nodes of the graph, may be joined (preferably after normalization) to another features space defined by the available information on the nodes. For instance, in the case of patents, a word space is defined by all words in all patent abstracts. Then, each patent vector may be extended by node features representing the connectivity of each patent in the citation graph. However, such a method is not well-designed for learning a predictive function. Indeed, a predictive function will have to assign a label to a new, not yet encountered node. While in such a setting having a new unlabeled point would require to add a new connection feature column to table (i.e. this new connection column will represent the connection between the new points and the already known points). This inevitably leads to retrain the model in order to take into account this new added column.

A more straightforward way to combine the topological information of a graph with information on its nodes is to turn the latter into a graph as well. In other words, we represent all the available knowledge with graphs. Plenty of different possibilities exist. For instance, we may infer a new graph directly from the table data set. Then, the adjacency matrix $\mathbf{A}$ may correspond to a weighted k nearest neighbors graph, or we may take a ratio r of the most relevant links to infer the graph.

The link weight may simply be the cosine computed between node vectors. In this case $a_{ij} = \cos(\mathbf{x}_i, \mathbf{x}_j)$. This is an affinity measure: high affinity means a cosine of 1.0, low affinity means two orthogonal vectors resulting in a cosine of 0.0 (we assume features are numerical and positive).

We may prefer to use the negative exponential of the cost matrix $\mathbf{W} = \exp \frac{-\mathbf{C}}{2\sigma_{\mathcal{G}}}$ where $c_{ij} = ||x_i - x_j||^2$ if there is an edge between vector $\mathbf{x}_i$ and $\mathbf{x}_j$ and ∞ otherwise. The width $\sigma_{\mathcal{G}}$ is fixed as the mean distance between adjacent nodes on the graph. The exponential is taken elementwise. The cost is the square of the distance one has to pay to transit from node i to node j. The exponential negative gives more importance to closer nodes than faraway (i.e. costly) nodes.

Recall that if the data is normalized, then the square of the distance between vector $\mathbf{x}_i$ and vector $\mathbf{x}_j$ is

$$\begin{aligned} d^2(i,j) &= ||\mathbf{x}_i||^2 + ||\mathbf{x}_j||^2 - 2\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j \\ &= 2 - 2\cos(i,j) = 2(1 - \cos(i,j)) \end{aligned}$$

Therefore,

$$\exp\left(-\frac{d^2(i,j)}{2\sigma_{\mathcal{G}}}\right) = \exp\left(-\frac{(1-\cos(i,j))}{\sigma_{\mathcal{G}}}\right)$$
$$= \exp\left(\frac{-1}{\sigma_{\mathcal{G}}}\right)\exp\left(\frac{\cos(i,j)}{\sigma_{\mathcal{G}}}\right)$$

In other words, having the cosine matrix we can easily compute $\mathbf{W}$. Once a graph is inferred from the data, graph-based algorithms can be applied to it.

If we have also at our disposal another graph showing the connectivity of the same data from another point of view, then we can combine the two information in, one, global graph. For instance, in the patent citation network, we find the original cited-citing network which link patents (i.e. the patent description is citing directly other existing patents). But we may also infer, as explained in previous paragraph, a weighted k nearest neighbors graph directly generated from the textual patents descriptions. In this inferred graph, patents are linked when their descriptions are similar. More precisely, each text description corresponds to a vector in the bag of words vector space formed by the patents descriptions corpus. Two patents are similar if the cosine (or the negative exponential of the cost) between the two associated word vectors is high. This way, a graph can be obtained directly from the information contained in a bag of words description of texts. We may wonder if working on such a representation of the data may be an interesting alternative to a standard table view model. Hence, in the next Section we will investigate how well graph based classifiers perform on a mined content graph compared to a standard classification algorithm (a linear SVM) that works directly on a bag of words representation.

7.2 First Experiment: Content Graph-based classifier vs. Content Feature-based classifier

In this Section, we infer graphs from textual information of two data sets and apply graph-based semi-supervised classification methods to them. The obtained results are compared to these obtained by a linear SVM classifier working directly on the table view model. We will compare them in terms of accuracy for several labeling rates.

First, we will do this for the well-know 20 newsgroups data set that was already benchmarked by Mitchell (1997); Joachims (1997) and others . This will enable to validate the implementation of the algorithms because the expected performance on this benchmark data set is well known. Afterwards, the algorithms will be applied to the large scale `U.S. patents` data set.

Data sets. *The 20 Newsgroups* data set is a collection of approximately 20,000 newsgroup documents, split across 20 different newsgroups. The collection became popular

set for experiments in text applications of machine learning techniques, such as text classification and text clustering. Each of the 20 newsgroups corresponds to a different topic. Some of the newsgroups are very closely related (e.g. comp.sys.ibm.pc.hardware and comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale and soc.religion.christian). Here is a list of the 20 newsgroups:

	News Groups
1	computer/windowsx
2	computer/pchardware
3	computer/machardware
4	computer/windows
5	computer/graphics
6	cryptography/general
7	electronics/general
8	forsale/general
9	medicine/general
10	motor/motorcycles
11	motor/autos
12	politics/general
13	politics/mideast
14	politics/guns
15	religion/general
16	religion/christian
17	religion/atheism
18	space/general
19	sport/hockey
20	sport/baseball

Table 7.1: NewsGroups classes

From these documents a k-nearest neighbors graph is inferred. For this purpose, each document is represented by a word vector whose components are computed with the well known *normalized tf.idf* factor. For each document we add to the content-graph k weighted links (i.e. undirected edges) that correspond to the k most relevant documents in terms of the cosine. The same operation is done for the `U.S. patents` abstracts in order to infer a k-nearest neighbors graph. The lucene package* was used to implement this task. Following Chapelle et al. (2006) k has been fixed to 30. Preliminary results for this experiment may be find in De Wagter (2010).

Classification methods. We compared a state of the art supervised categorization technique to several graph-based semi-supervised algorithms introduced in previous chapters. As supervised algorithm we used: (1) the support vector machine (SVM) with a linear kernel. For the graph-based techniques we tested the top leading algorithms of the previous Chapter: (2) the Sum-over-Paths (SoP), (3) the normalized random walk with restart (NRWR) and (4) the biased $\mathcal{D}$WALK.

*http://lucene.apache.org/

Experiments. As before, the classification accuracy will be reported for several labeling rates (1, 5, 10, 20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The remaining nodes are used as test data. For each considered labeling rate, 10 random node label deletions (test sets) were performed (10 runs), on which performances were averaged. For each run, a internal 10-fold cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (see the first experiment in Section 6.3.1 for details). The performance on each run is assessed with the hyper-parameter tuned during the 10-fold cross-validation.

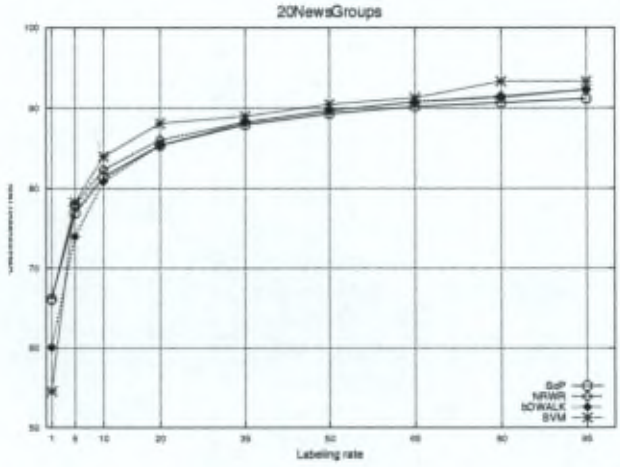


Figure 7.1: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for increasing labeling rates 1, 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, bNRWR, aNRL, bDWALK and SVM classification methods. The graphs show the results obtained on an inferred weighted 30 nearest neighbors graph build from the `20 NewsGroups` data set. There are 20 classes in total, the majority class proportion represents 5% of the data. For a low labeling rate (i.e. $< 5\%$) graph-based algorithms obtain better accuracy than SVM.**

Results and discussion. As expected, the state-of-the art supervised SVM algorithm obtains the best accuracy when the training set is large enough (i.e. more than 5% of the data, see Figure 7.1 and 7.2). However, the graph-based semi-supervised techniques are competitive, also for high labeling rates. Indeed, the loss is only between 1 and 3% depending of the labeling rate and of the graph-based algorithm considered. What is more interesting, are the results when we lack training data (i.e. with a low labeling rate $\leq 1\%$, $\leq 0.1\%$). In this case, semi-supervised techniques perform better. Indeed, the loss of accuracy observed between the SVM and both bDWALK and the NRWR is more than 5%. Notice that on the `U.S. patents` data set we also consider low

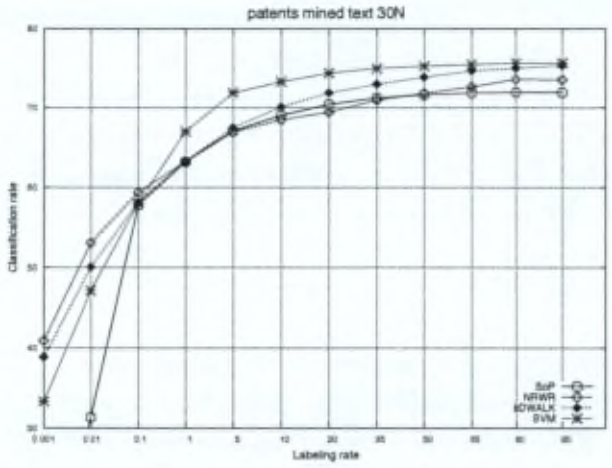


Figure 7.2: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for increasing labeling rates 1, 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and SVM classification methods. The graphs show the results obtained on an inferred weighted 30 nearest neighbors graph build from the `U.S. patents` abstracts. There are 6 classes in total, the majority class proportion (i.e. Mechanical category) represents 22% of the data.**

labeling rate $\leq 0.1\%$ since the data set is very large (i.e. more than 2.2 million of documents).

7.3 Second Experiment: Graph-based Text Categorization

As just discussed, text categorization may also be addressed by first inferring a sparse network (e.g. a k weighted nearest neighbor graph) and by then applying a graph based classification algorithms to it. One interesting question appears in the case where we have a citation graph associated to the document corpus. In that case, it would be interesting to know if graph-based classification methods work better on the inferred content graphs from text, or on the citation graphs. In the previous Chapter, we assess the performance of graph-based classification methods only on citation graphs. Therefore, in this section, we will infer sparse content graphs from six already encountered data sets for which textual information is available: The four `WebKb`, the `CoRA` and the `ticker` data set, i.e. the `industry-pr` data set (see Section 5.3 for a complete description of these data sets). Recall, in the citation graphs links have been inferred from a real interaction between two nodes in the original network. Indeed, the `WebKB` data sets consist of sets of web pages from four computer science departments where

links stand for co-citation between two web pages. This time, however, we propose to extract the text associated to each web page of the entire data set in order to collect one corpus for each WebKB department. Then, as previously, we simply infer a k weighted nearest neighbors graph from the textual information. The `CoRA` data set comprises computer science research papers. While previously we used the citation graph, this time, we collect the the abstracts of papers and infer a sparse graph. The ticker (i.e. industry-pr) data set can also be subject to the same process since every node represents a newswire press release. We extracted the graphs for 5, 10, 20 and 50 nearest neighbors.

Classification methods. We apply the top semi-supervised graph-based methods on the inferred graphs: (1) the sum-over-paths (SoP), (2) the normalized random walk with restart (NRWR), (3) the biased $\mathcal{D}$WALK and (4) the normalized regularized laplacian (NRL).

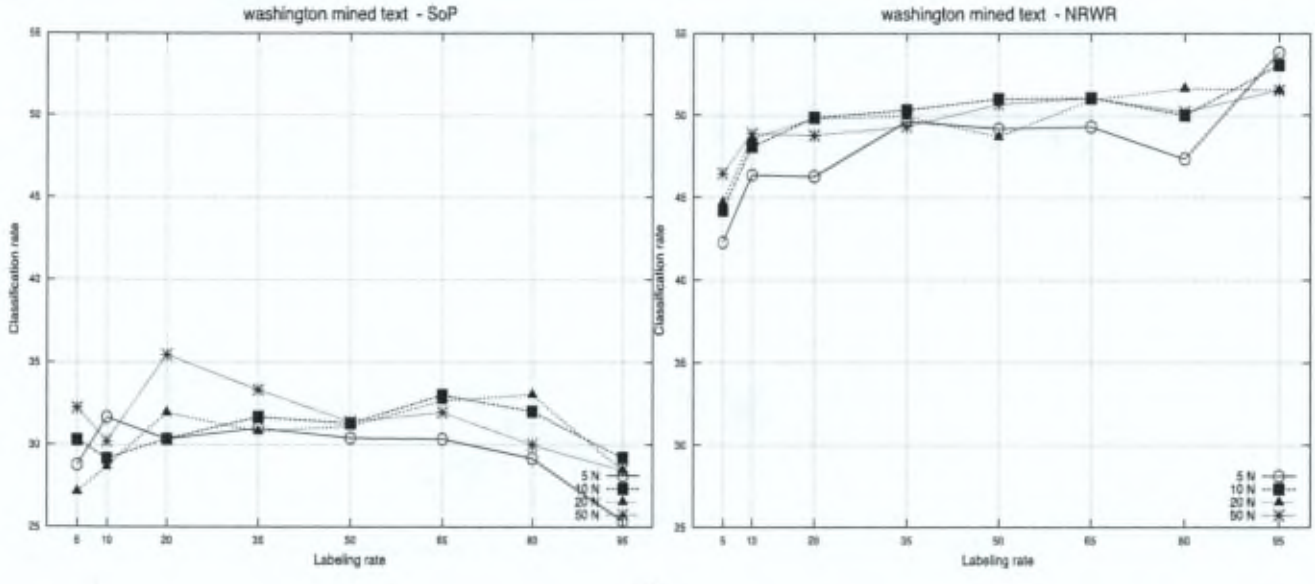
Experiments. As before, the classification accuracy will be reported for several labeling rates (5, 10, 20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The labels of the remaining nodes were removed and used as test data. For each considered labeling rate, 10 random node label deletions (test sets) were performed (10 runs), on which performances were averaged. For each run, an internal 10-fold cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (see the first experiment in Section 6.3.1 for details). The performance on each run is assessed on the remaining unlabeled nodes with the hyper-parameter tuned during the 10-fold cross-validation.

Results and discussion. Whatever the method and the data set, the inferred text graphs obtained worse results (Figures 7.3, 7.4, 7.5, 7.6, 7.7 and 7.8) in comparison to their citation counterpart graphs. The difference in terms of accuracy is very significant in favor of the citation graphs (Table 7.2 and 7.3). This is an important result favoring citation graphs for classification task.

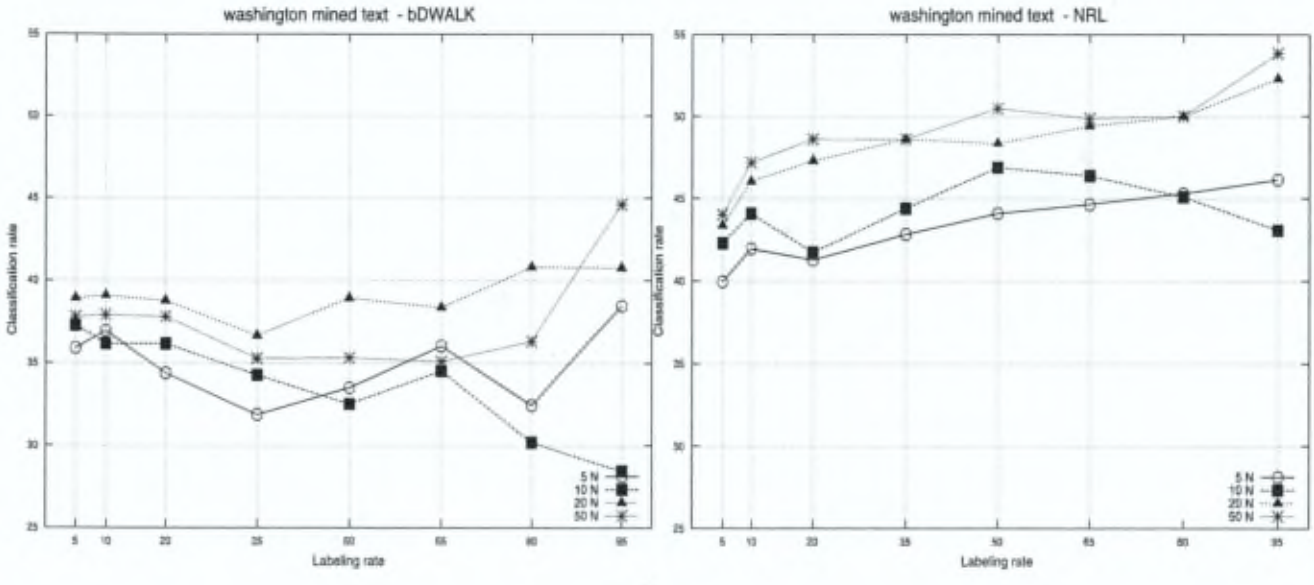
	b$\mathcal{D}$WALK	
	Citation graph	Content-based graph
`texas`	64.11	62.81
`wisconsin`	65.83	35.10
`washington`	63.56	35.90
`cornell`	49.69	52.59
`CoRa`	76.25	17.36
`industry-pr`	31.38	31.07
`U.S. patents`	80.74	67.51

Table 7.2: **Accuracy results for 5% labeling rate for the b$\mathcal{D}$WALK approach for seven data sets on the citation graph and the 5 nearest neighbors graph inferred from textual information. The accuracy on the citation graphs is clearly better.**

The NRWR and b$\mathcal{D}$WALK achieve the best results according to a significance signed

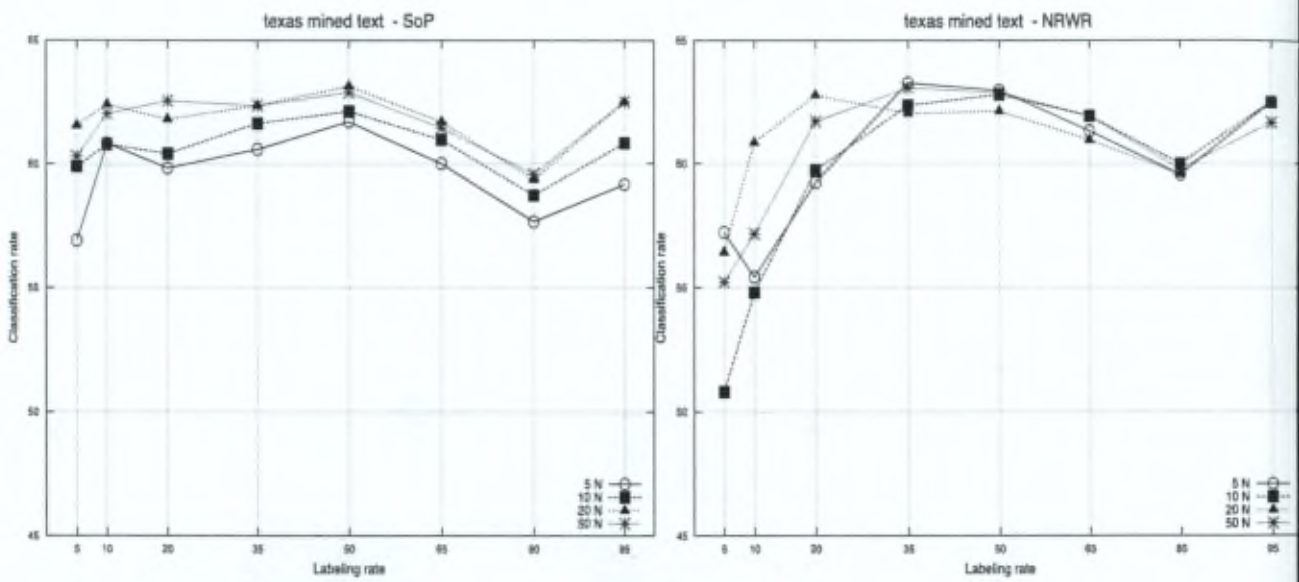


(a)

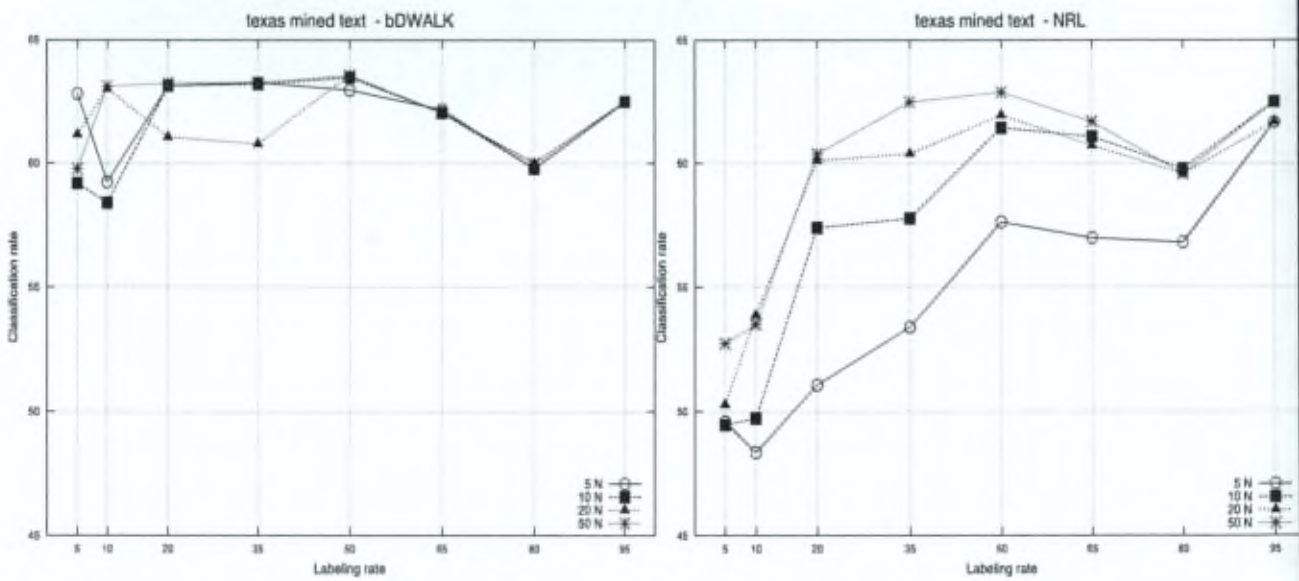


(b)

Figure 7.3: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the 5, 10, 20 and 50 weighted nearest neighbors graphs mined from the `WebKB washington` data set (see Macskassy, 2007).**

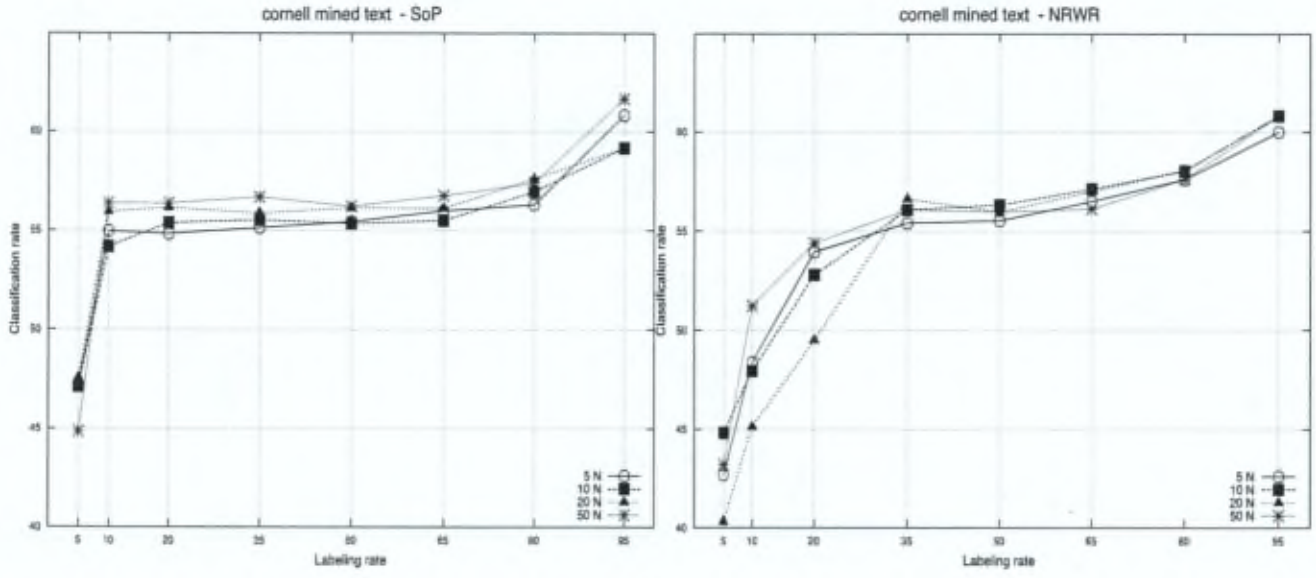


(a)

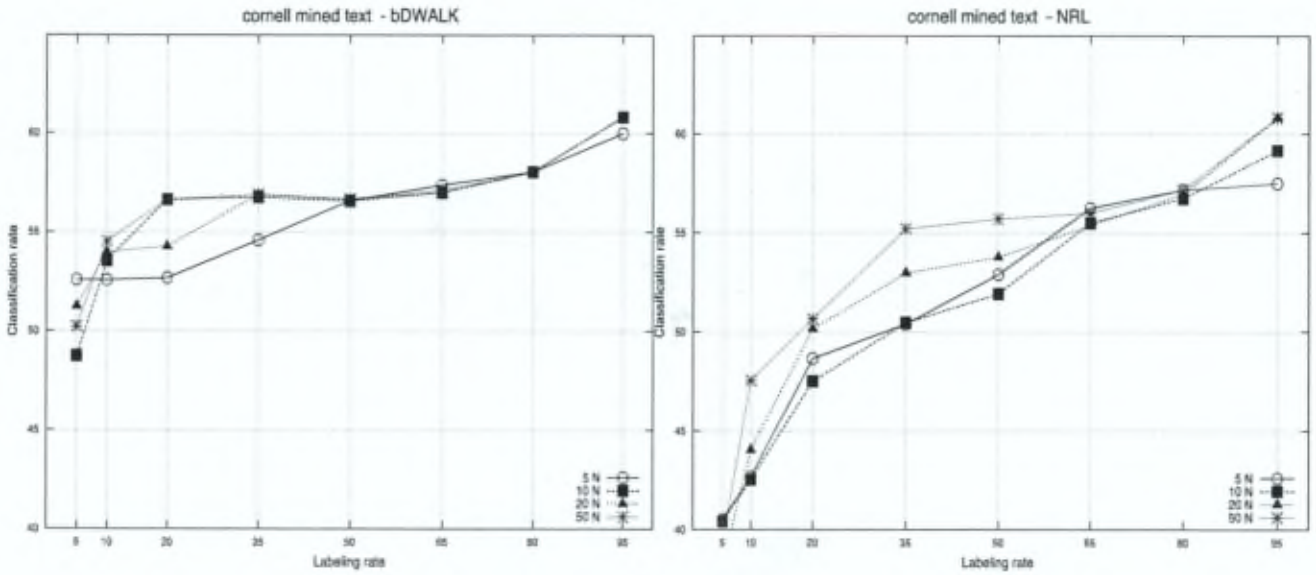


(b)

Figure 7.4: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, bDWALK and NRL classification methods. The graphs show the results obtained on the 5, 10, 20 and 50 weighted nearest neighbors graphs mined from the `WebKB texas` data set (see Macskassy, 2007).**

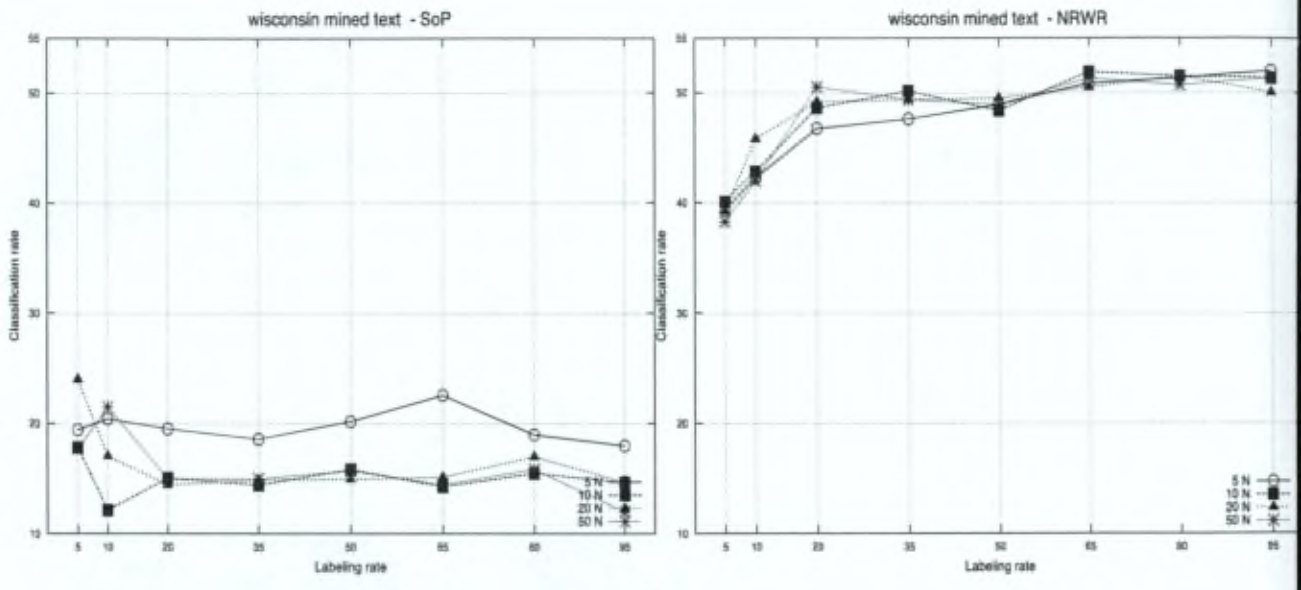


(a)

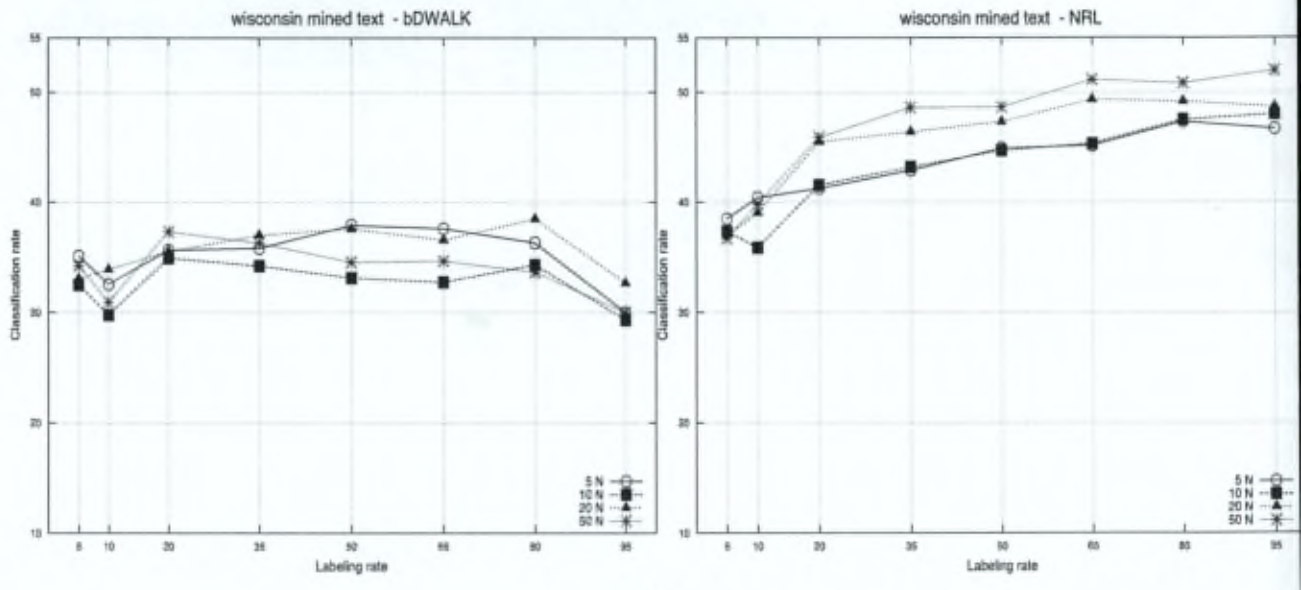


(b)

Figure 7.5: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, bDWALK and NRL classification methods. The graphs show the results obtained on the 5, 10, 20 and 50 weighted nearest neighbors graphs mined from the `WebKB cornell` data set (see Macskassy, 2007).**



(a)



(b)

Figure 7.6: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, bDWALK and NRL classification methods. The graphs show the results obtained on the 5, 10, 20 and 50 weighted nearest neighbors graphs mined from the `WebKB wisconsin` data set (see Macskassy, 2007).**

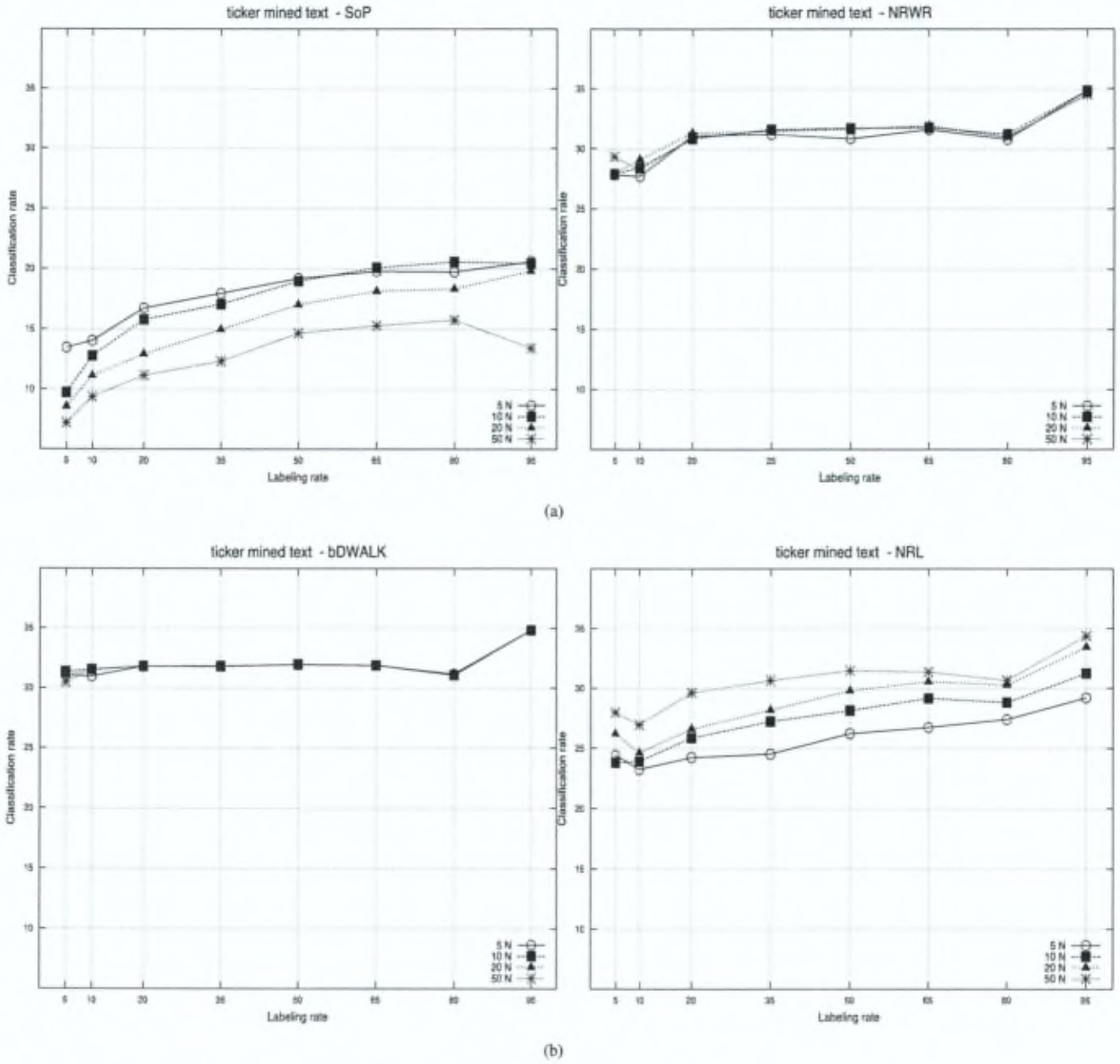
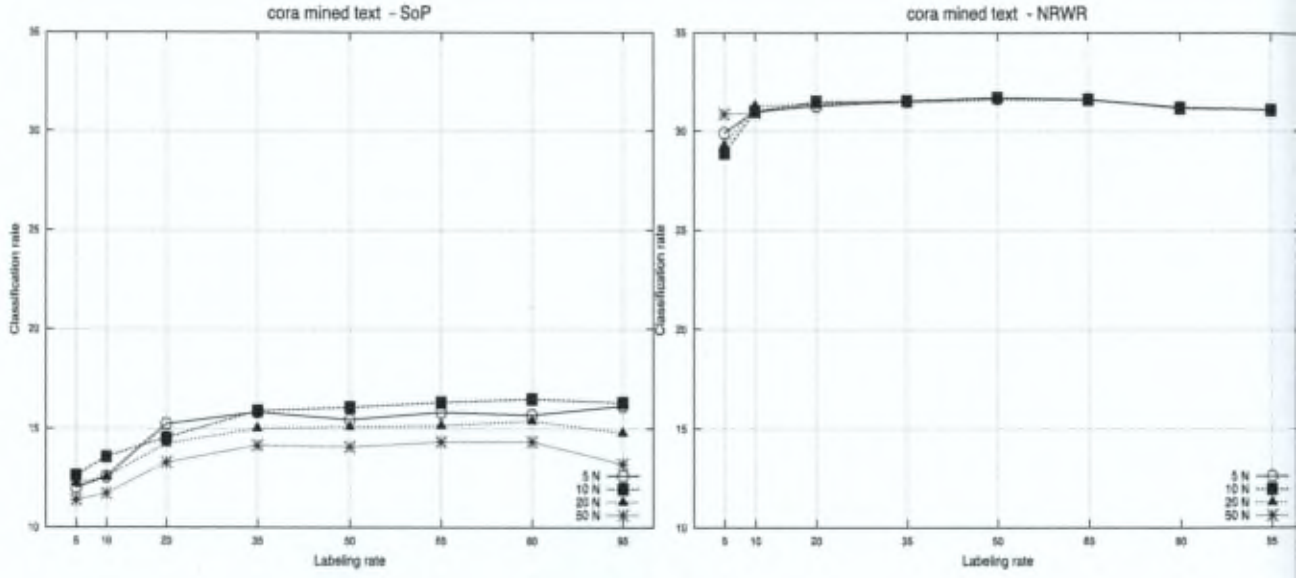
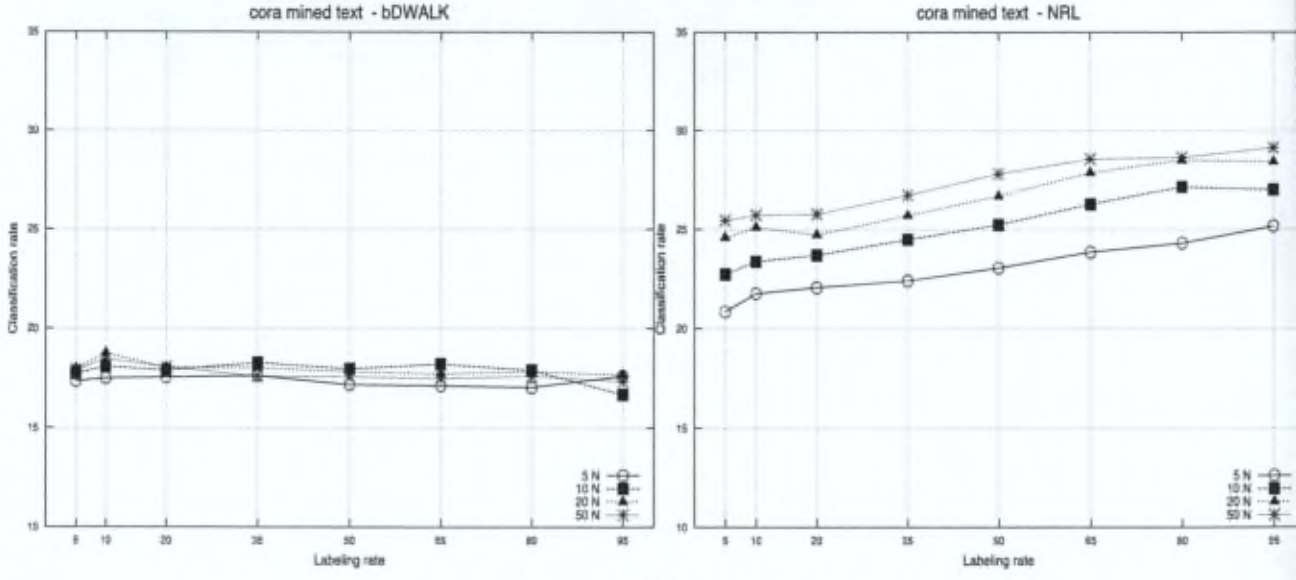


Figure 7.7: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the 5, 10, 20 and 50 weighted nearest neighbors graphs mined from the `industry-pr` data set (see Macskassy, 2007).**



(a)



(b)

Figure 7.8: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the 5, 10, 20 and 50 weighted nearest neighbors graphs mined from the CoRA data set (see Macskassy, 2007).**

	NRWR	
	Citation graph	Content-based graph
texas	63.27	57.23
wisconsin	65.83	39.33
washington	63.24	42.28
cornell	50.06	42.68
CoRa	76.25	29.89
industry-pr	32.27	27.83
U.S. patents	81.55	66.99

Table 7.3: **Accuracy results for 5% labeling rate for the NRWR approach for seven data sets on the citation graph and the 5 nearest neighbors graph inferred from textual information. The accuracy on the citation graphs is clearly better.**

test — p-value < 0.01 — for 5% labeling rate on the 5 and 50 neighbors data sets and in terms of accuracy (Table 7.4 for statistical sign tests). This confirms that these two algorithms are well suited for semi-supervised learning since they already obtain the best results on the citation graphs in the previous chapter (see Section 6.3.1 and Section 6.3.2).

One may wonder what is the best number of neighbors k to extract when building the data set in order to improve the classification rate. Intuitively, for a low labeling rate it is probably better to take more neighbors. Indeed, the 5 neighbors classifiers accuracy are significantly worse for a low labeling rate of 5% 20 times out of 24 (4 graph-based algorithms applied to 6 data sets), hence, suggesting to take more neighbors into account. However, in the case of a higher labeling rate there is no evidence to prefer some number of neighbors to another.

Considering this fact, another interesting question is to analyze the robustness of the different algorithms with respect to the number of neighbors. In particular, we would like to identify the method that is the least sensitive to variations in k. Hence, we report in Table 7.5, for each method and for each data set, the averaged RMSE between the 5 neighbors accuracy curve and the 50 neighbors accuracy curve. The lower the RMSE, the more robust the method. The right-hand side column of this Table corresponds to the mean of the other columns. The two methods NRWR and bDWALK clearly are less sensitive to k and thus more robust.

If we compare the obtained results on content inferred graphs to their citation counterpart graphs (Section 6.3.1 and Section 6.3.2), we observe that the classification accuracy is always significantly better on the citation graphs (Table 7.2 and 7.3). This suggests that citation links are more appropriate for diffusing labels than mined text links. Generally, a citation occurs only between two documents in the same class. While, in the mean time, documents of different classes may still be similar. This means that citation graphs probably fulfill more the local consistency assumption that two neighbor nodes should belong to the same class. Therefore, in the remainder, when combining

citation links with mined text links, we will try to make better use of citation links in order to improve semi-supervised classification accuracy.

Algorithm	SoP	NRWR	b$\mathcal{D}$WALK	NRL
SoP	—	= (3), < (9)	< (12)	< (8), > (4)
NRWR	—	—	> (6), = (4), < (2)	> (12)
b$\mathcal{D}$WALK	—	—	—	> (12)

Table 7.4: **Compilation of statistical sign tests computed for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. A signed test was performed on each of the 12 mined text data sets: the four `WebKB`, `ticker` and `CoRA` for 5 and 50 neighborsdata sets. The test is based on the results of 10 runs of the semi-supervised classification task with a low labeling rate of 5%. Each entry in the table shows the number of times that the row method is significantly (i.e. p-value < 0.01) better (>), equivalent (=) or worse (<) respectively than the column method. For instance, NRWR performs significantly better than b$\mathcal{D}$WALK on 6 data sets.**

Algorithm	`texas`	`wisconsin`	`washington`	`cornell`	`ticker`	`CoRA`	*Mean*
SoP	6.45	13.03	7.69	3.95	15.25	4.72	*8.51*
NRWR	3.77	4.42	6.98	3.15	1.89	0.97	***3.53***
b$\mathcal{D}$WALK	4.96	5.80	9.20	5.57	0.82	1.48	***4.63***
NRL	16.2	12.19	16.75	8.44	13.37	12.18	*13.19*

Table 7.5: **Averaged RMSE computed on 10 runs for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The results are reported for the four `WebKB` data sets, the `ticker` (i.e. `industry-pr`) data set and the `CoRA` data set. The RMSE is computed between the 5 neighbors curve and the 50 neighbors curve reported in Figures 7.4, 7.3, 7.6, 7.5, 7.7 and 7.8. The NRWR and b$\mathcal{D}$WALK are more stable than other methods in terms of number of neighbors.**

7.4 Third Experiment: Combining Citation Graph and Content Graph

It may happen that different sources of information are available: A citation network (for instance: the patent citation network, etc.), and a set of features on independent observations (for instance, patents abstracts and other relevant features like the date of submission, the owner, etc.). In this case, we can combine the different available sources of information. More precisely, let us detail two different use cases:

1. We have at our disposal a network (i.e. citation graph) and we want to use external available information in order to improve the classification accuracy. For instance, in biology, a protein-interaction network can be combined with diverse information on proteins taken from databanks for improving classification accuracy.

2. We have at our disposal a set of independent descriptive objects to classify. In this second case, available external resources, often structured as a graph, may be used in order again to improve the classification accuracy of the independent objets. More concretely, currently, one hot topic in data mining and information retrieval is the use of external semantic resources or social networks in order to improve performance of state-of-the-art techniques (see, e.g., Hu et al., 2009). For instance, we may want to improve the classification score through the use of external graph semantic resources like Wikipedia, WordNet, etc. (see, e.g., Lehwark et al., 2010). Here, we will consider as external resource an external citation graph.

The goal is this section will be to investigate if we can improve the accuracy results by combining content and citation information. In both cases there are different ways to combine the two sources of information. The fusion of various sources of knowledge has been an active subject of intensive research since more than three decades (see for some review references Cooke, 1991; Genest and Zidek, 1986; Jacobs, 1995). It has recently been successfully applied to the problem of classifiers combination or fusion (see for instance Kittler et al., 1998). Many different approaches have been developped for experts opinions fusion, including weighted average (see for instance Cooke, 1991; Jacobs, 1995), Bayesian fusion (see,e.g., Cooke, 1991; Jacobs, 1995), kernel fusion (see the Ph.D. thesis of Yu) majority vote (see,e.g., Chen and Cheng, 2001; Kittler and Alkoot, 2003; Lad, 1996), models coming from incertainty reasoning: fuzzy logic, possibility theory (see,e.g., Klir and Folger, 1988; Dubois et al., 1999), standard multivariate statistical analysis techniques such as correpondence analysis (Merz, 1999), maximum entropy modeling (see,e.g, Levy and Delic, 1994; Myung et al., 1996; Saerens and Fouss, 2004).

To fusion data, graphs also constitute another way to structure all information together. Citation graphs and content graphs (i.e. k nearest neighbors graphs build from text information) may always be joined in one global graph. By doing that, we create new paths into the network that may hopefully help the classifier in his original task (Figure 7.9). This is the solution that will be investigated is this work.

After fusion, three different types of nodes appear: citation-only nodes (1), citation and document nodes (2) and document-only nodes (3). Citation-only nodes concern nodes for which we do not have the text; document-only nodes concern documents that are never cited by any document. Document nodes that appear in the citation graph (i.e. (2)) are bridge nodes between (1) and (3), hence creating new label propagation paths from citation nodes to document nodes. Intuitively, the idea will be to diffuse labels in that direction since our previous experiments showed us that citation labels are quite accurate.

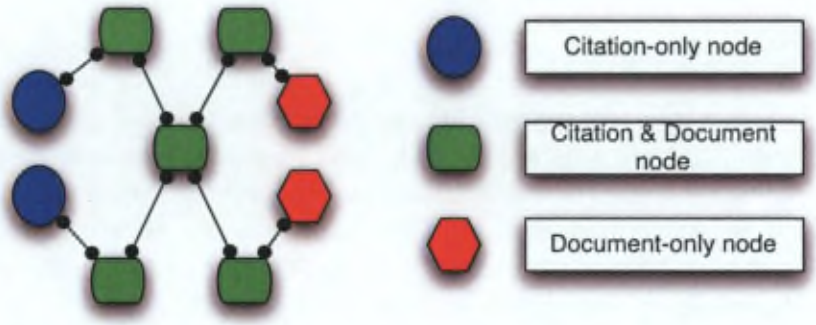


Figure 7.9: **This graph represents the merging of two graphs: a citation graph and a k-nearest neighbors graph derived from textual information. After fusion, three different kinds of nodes appear: citation-only nodes (1), citation and document nodes (2) and document-only nodes (3). Citation-only nodes concern nodes for which we do not have the text, document-only nodes concern documents that are never cited by any other document. Document nodes that appear in the citation graph are brigde nodes between (1) and (3).**

In the sequel, we will present two experiments. The first one will investigate to which extend the accuracy can be improved when combining a citation graph to an inferred graph from textual information on its nodes (i.e. a content graph). The goal is to assess if we can improve classification accuracy for citation nodes. This will be referred to Experiment 1. The other way around, the second experiment will investigate to which extend the accuracy of semi-supervised text based categorization can be improved by merging the content inferred graph with an external citation graph (as explained on Figure 7.9). Here, the goal will be to assess if we can improve text categorization accuracy on document-only nodes of a global merged graph, in comparison to previous experiments only working on content inferred graphs (see previous experiment Section

7.3). This will be referred to Experiment 2.

Classification methods. For the two experiments, we apply the best semi-supervised graph-based methods: (1) the sum-over-paths (SoP), (2) the normalized random walk with restart (NRWR), (3) the biased $\mathcal{D}$WALK and (4) the normalized regularized laplacian (NRL).

Experiment 1 - methodology. As before, the classification accuracy will be reported for several labeling rates (5, 10, 20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The assessment methodology is exactly the same as in Section 6.3.1. The performance on each run is assessed on the remaining unlabeled citation nodes (i.e. nodes that appear in the citation graph) with the hyper-parameter tuned during the 10-fold cross-validation, except for the large `U.S. patents` where the hyper-parameters have been directly tuned with the same values as these used in a previous experiment because of computing time issues (Section 6.3.2). In this first experiment, we report for each data set the results obtained on a combined graph. The combined graph is the union of the citation graph and a subgraph of the 50 nearest neighbors inferred documents graph. The subgraph is obtained by only keeping the most relevant links. We report results for several link selection rates (0.0, 0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1.0). The link selection rate is the ratio of all relevant links that are added to the original citation graph. Hence, 0.0 indicates the original graph only, and 1.0 the union of both graphs. After selection, the weights of the selected links are set to 1. In other words, it is as if missing citations are inferred from text analysis and are added to the citation graph. Note that for the NRL method the results are not reported for low link selection rates; this is because the NRL algorithm requires the graph to be connected. Link selection rates < 1.0 may result in a possibly disconnected graph after union.

Results and discussion. After analysis, it appears that the addition of text mined links into the original citation graph does not help to classify more accurately (Figures 7.10, 7.11, 7.12, 7.13, 7.14, 7.15 and 7.16). Indeed, whatever the links selection rate considered, no significative increase in accuracy is observed. Depending on the method and the labeling rate, the results are sometimes slightly better and sometimes slightly worse. As already mentioned in the previous experiment, this is due to the fact that better results are obtained with the citation graphs than with the content-based graphs. Hence, combining these two informations does not help to improve the performance of the already more accurate information: the citation graph. In this case, it seems that by adding text-based links to the citation graph, we are not enforcing the weight of the good diffusion links (i.e. original citation links). Notice also that the b$\mathcal{D}$WALK method is less sensitive (i.e. more robust with respect to the link selection rate than the other methods) and, overall, obtain good performance. Indeed, the variance between the different selection rate curves is much less for the b$\mathcal{D}$WALK than for the other methods.

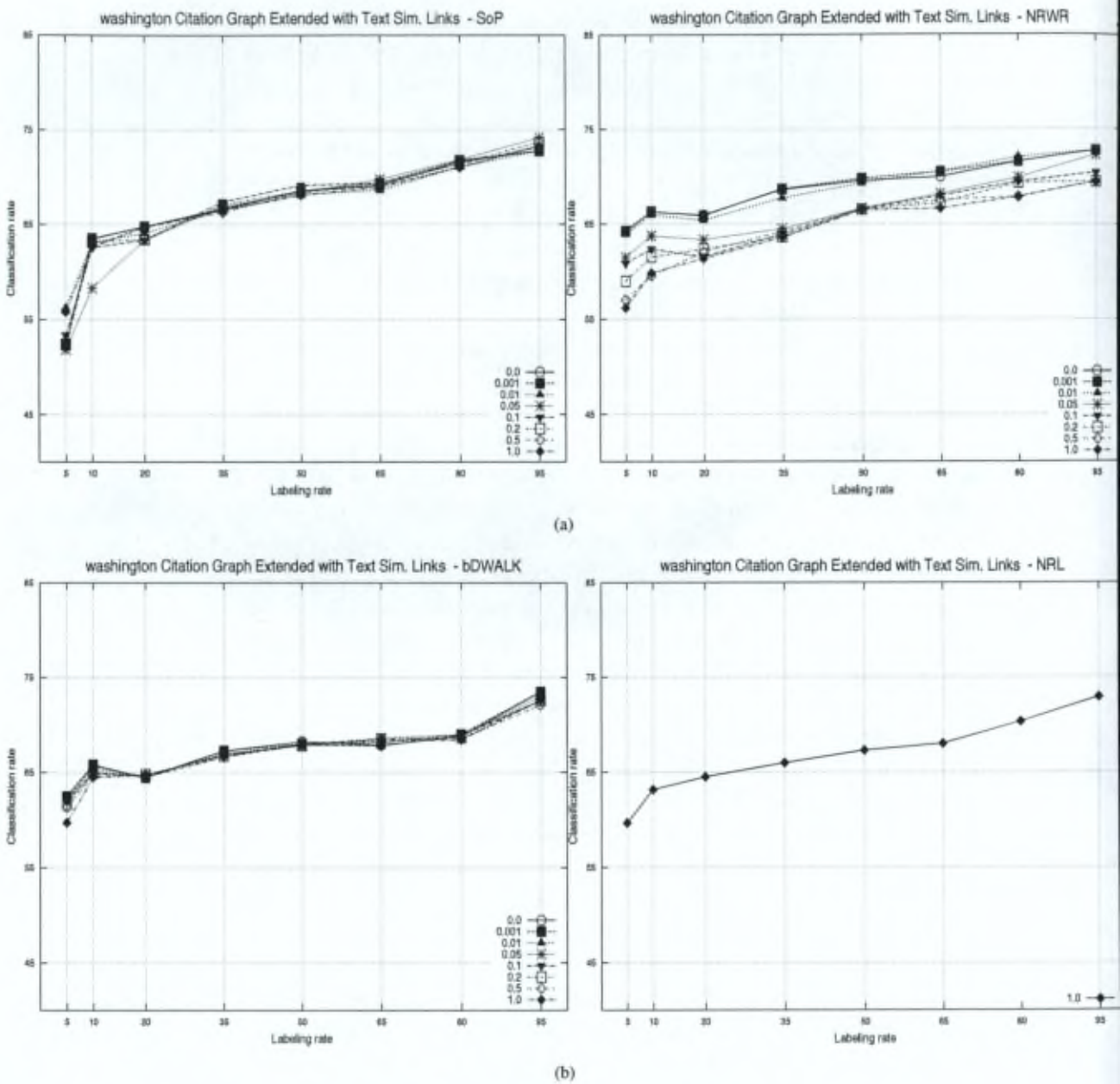


Figure 7.10: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `WebKB washington` data set.**

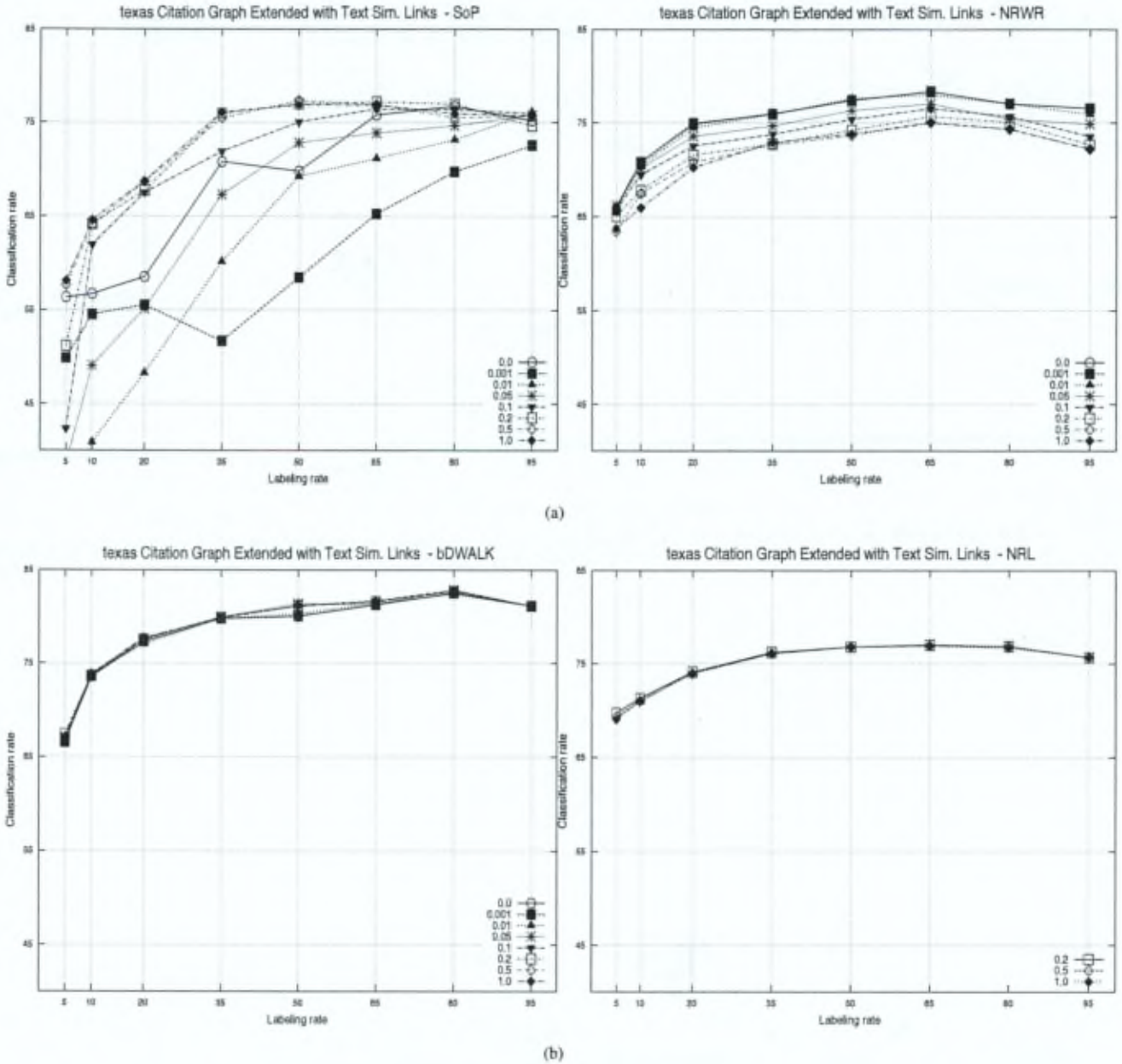


Figure 7.11: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, bDWALK and NRL classification methods. The graphs show the results obtained on the `WebKB texas` data set.**

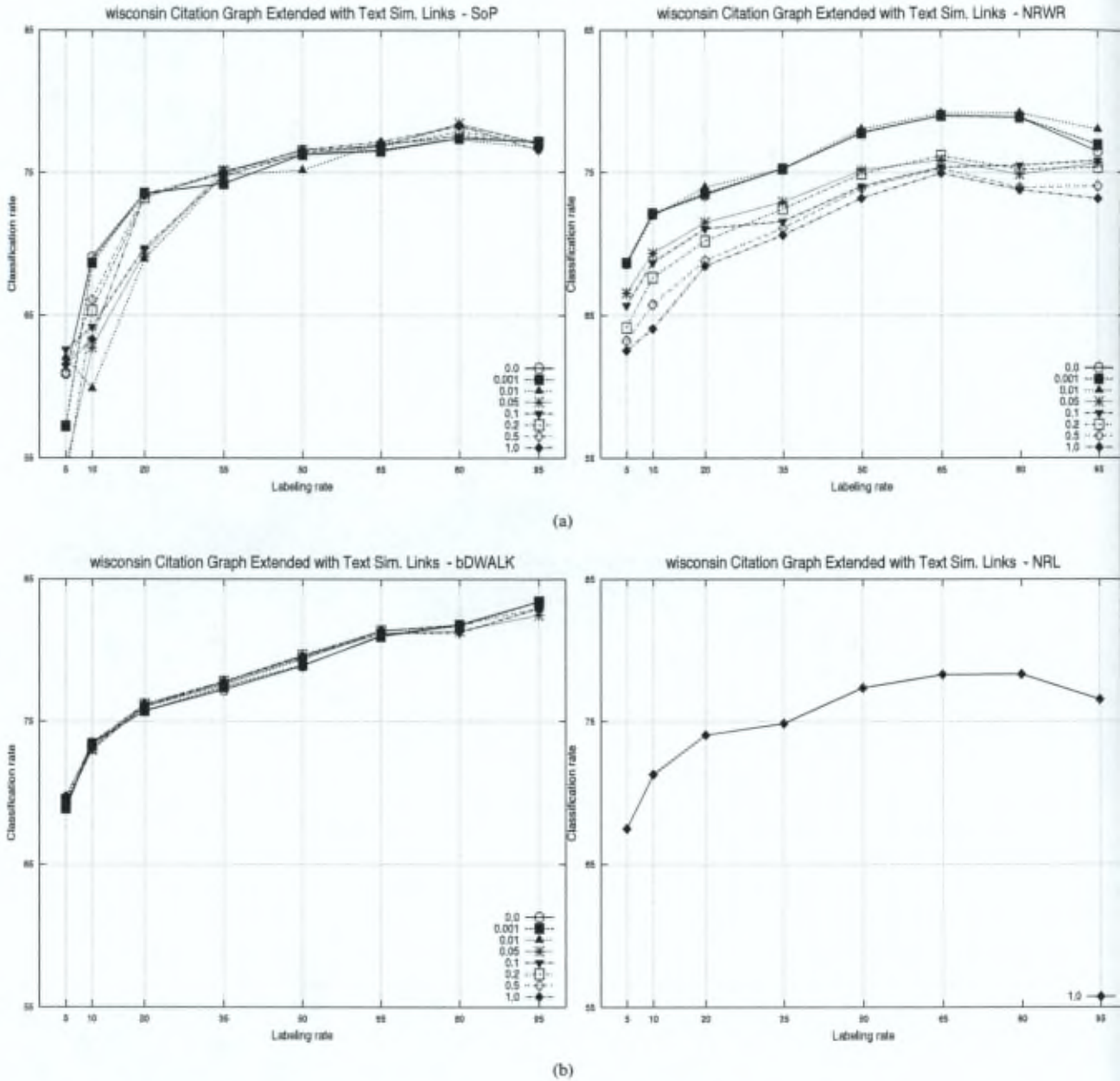
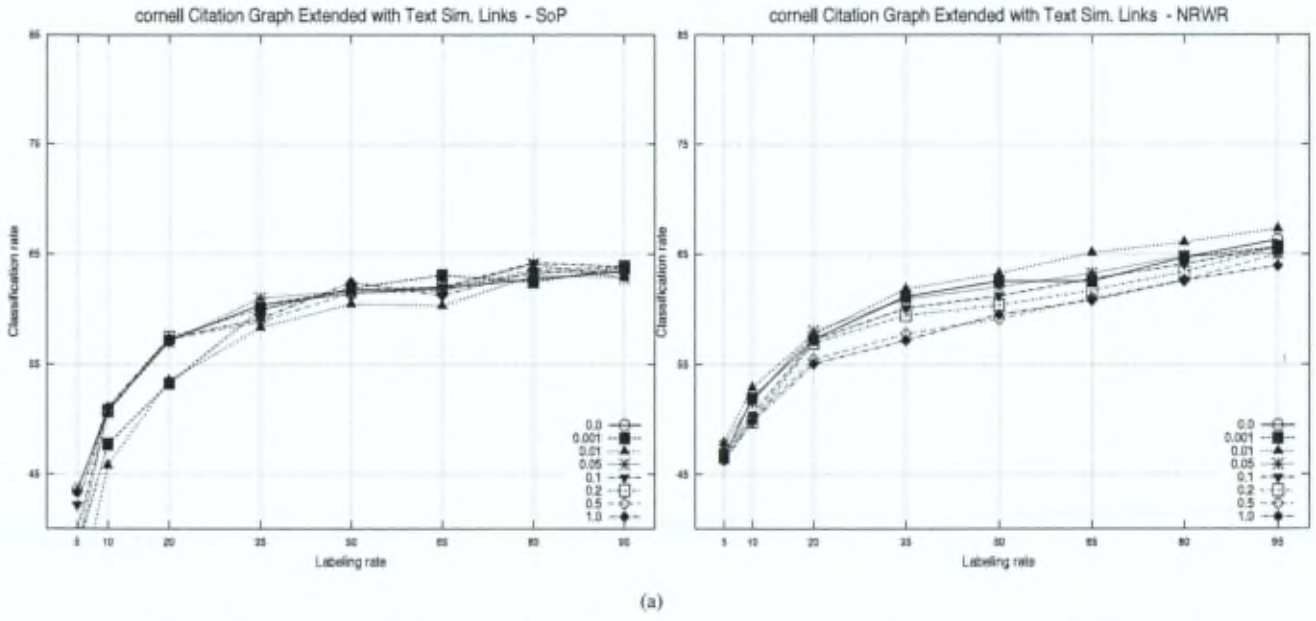
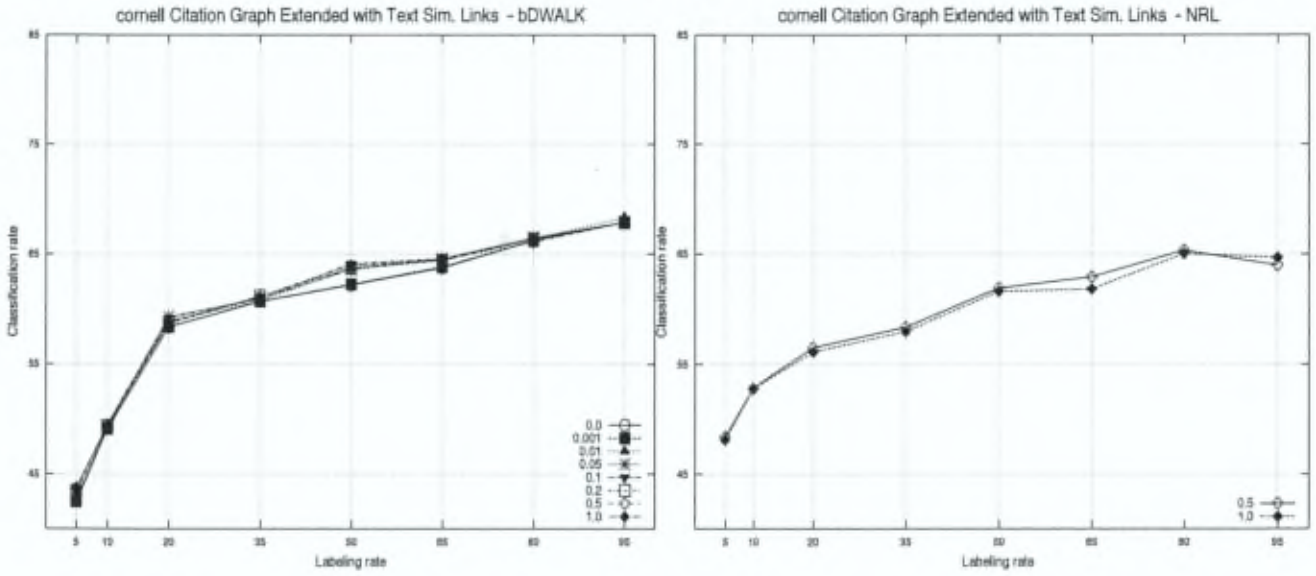


Figure 7.12: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, bDWALK and NRL classification methods. The graphs show the results obtained on the `WebKB wisconsin` data set.**

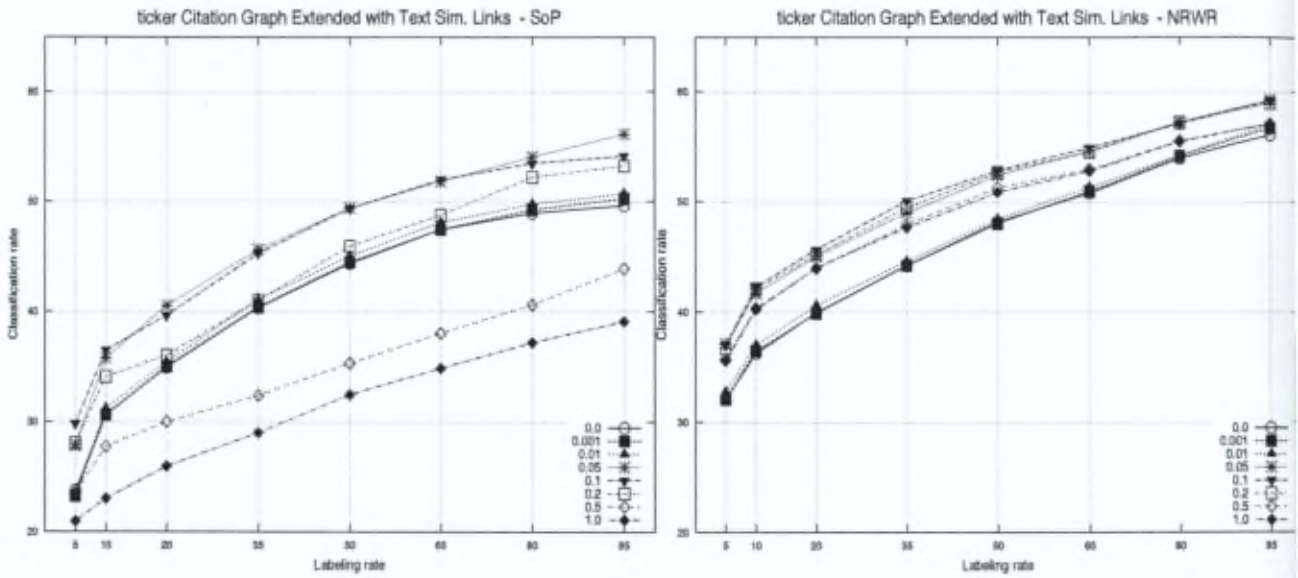


(a)

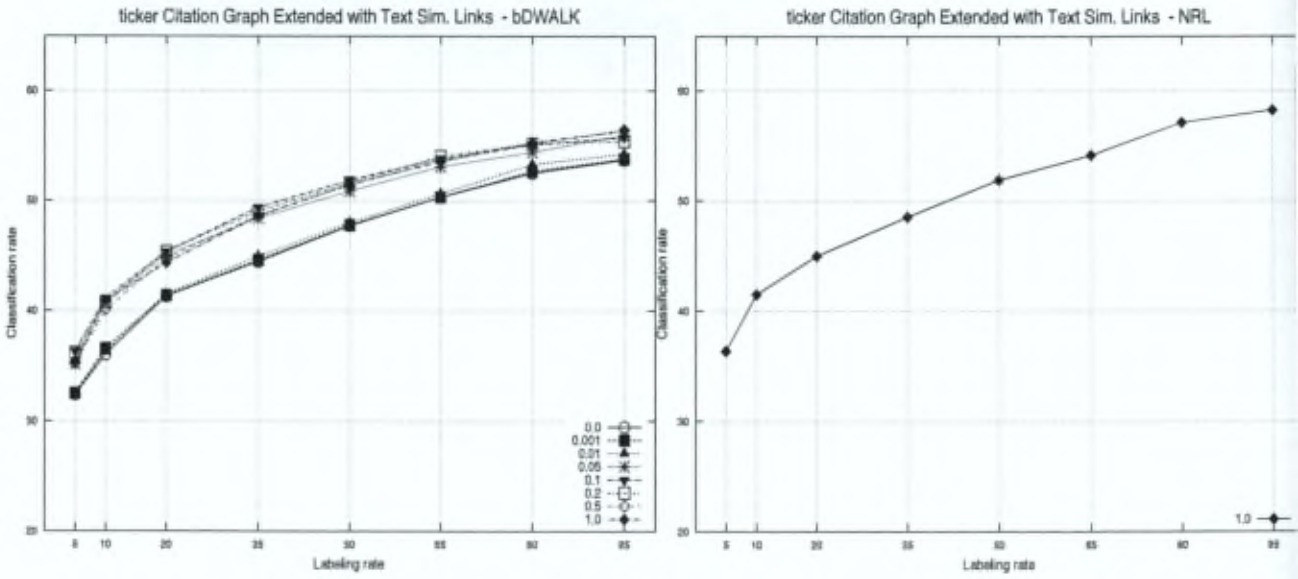


(b)

Figure 7.13: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `WebKB cornell` data set.**

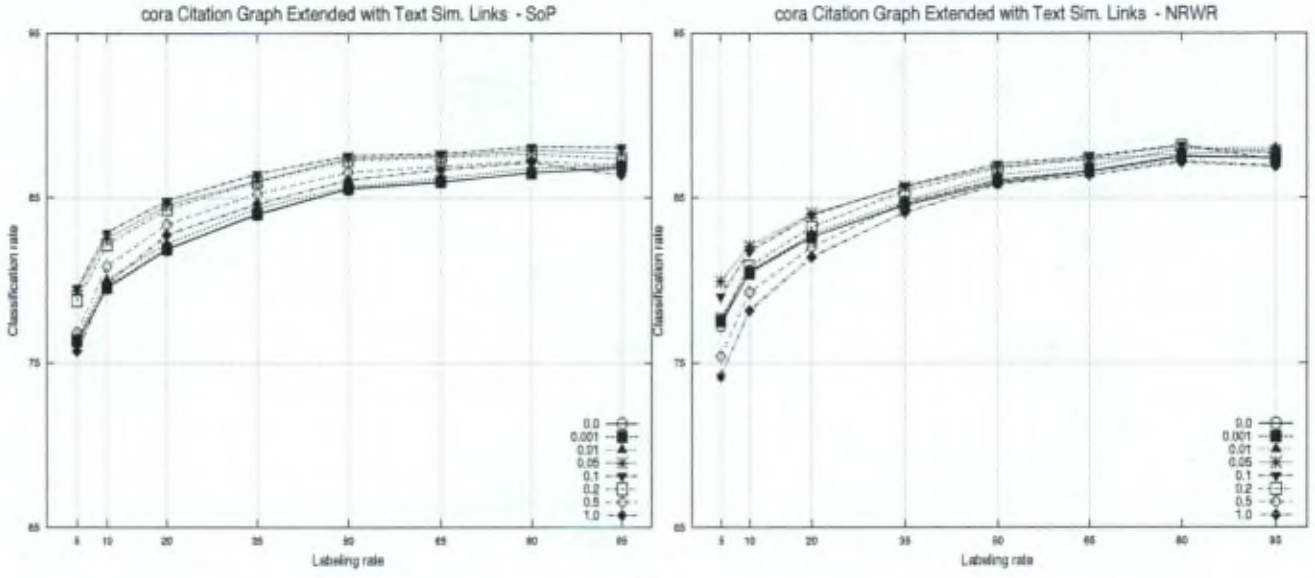


(a)

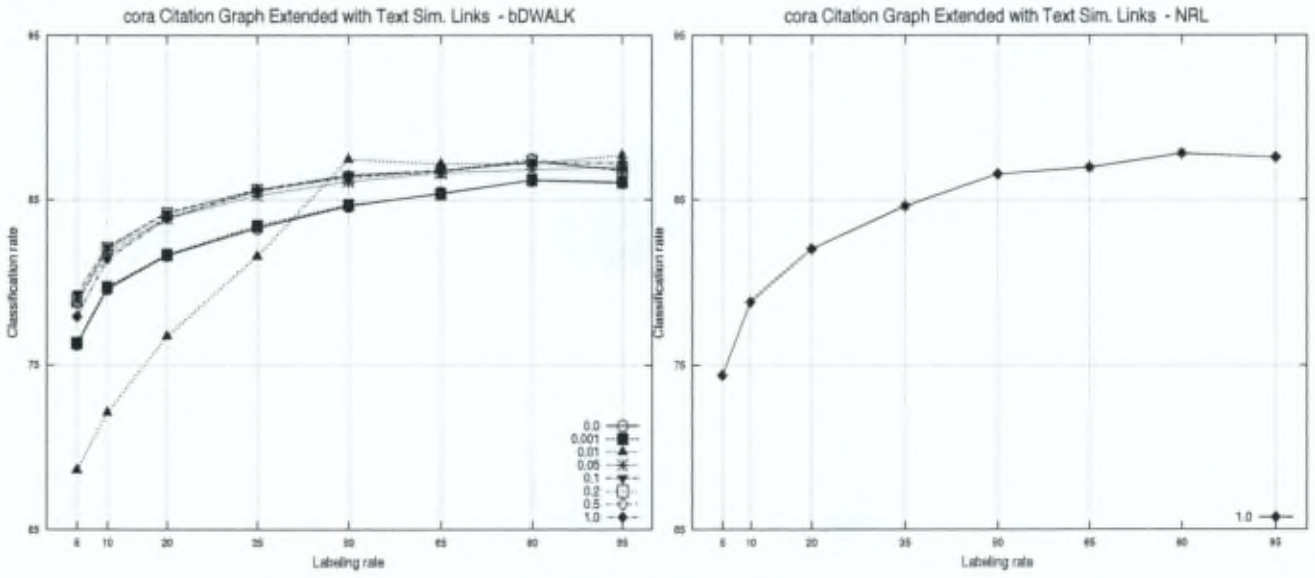


(b)

Figure 7.14: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `industry-pr` data set.**

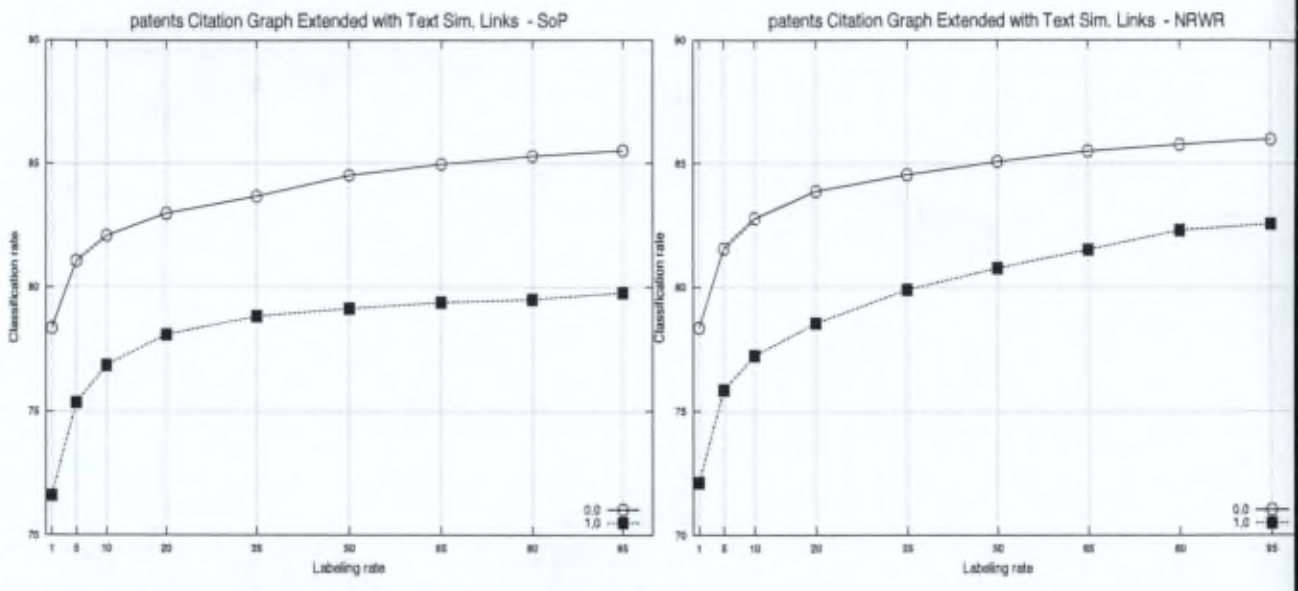


(a)

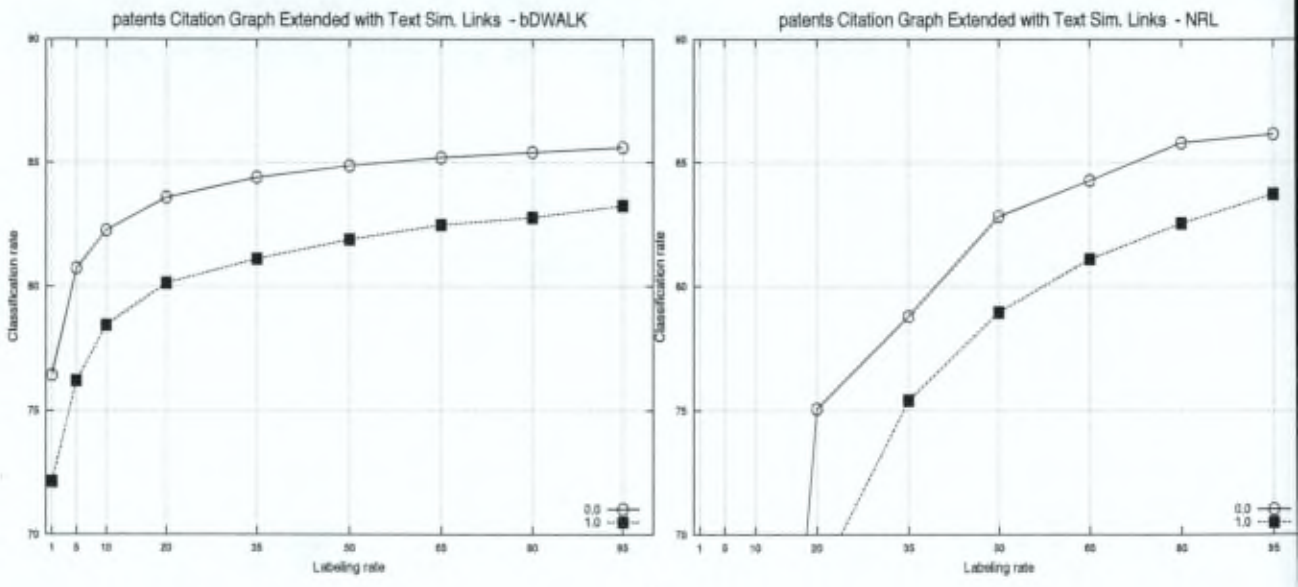


(b)

Figure 7.15: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, bDWALK and NRL classification methods. The graphs show the results obtained on the CoRA data set.**



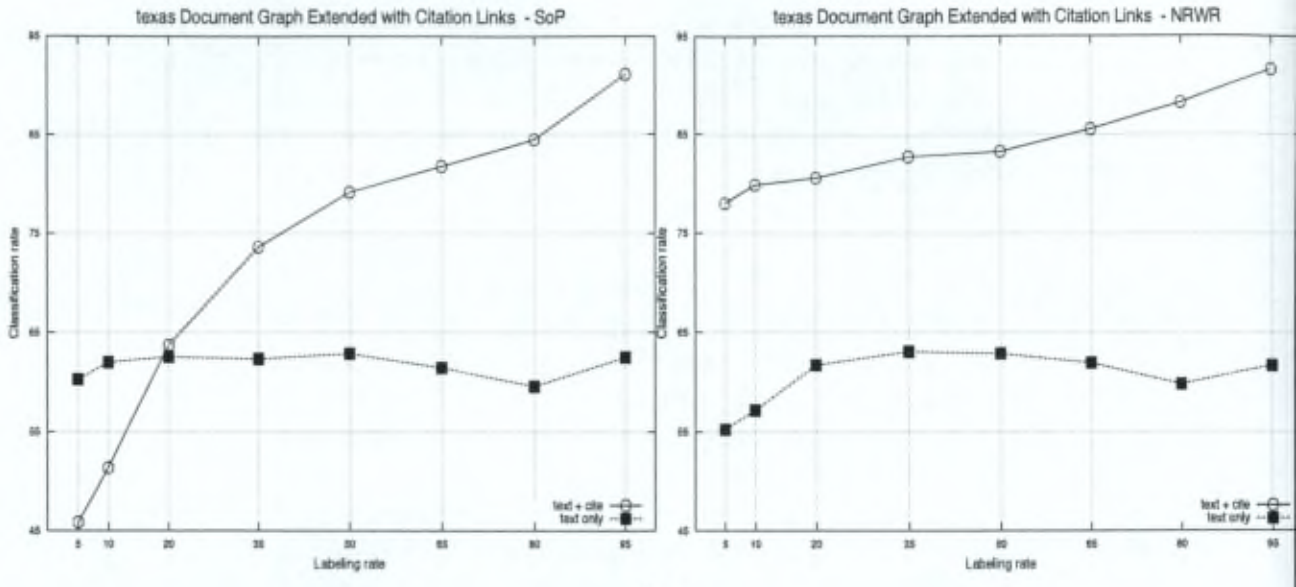
(a)



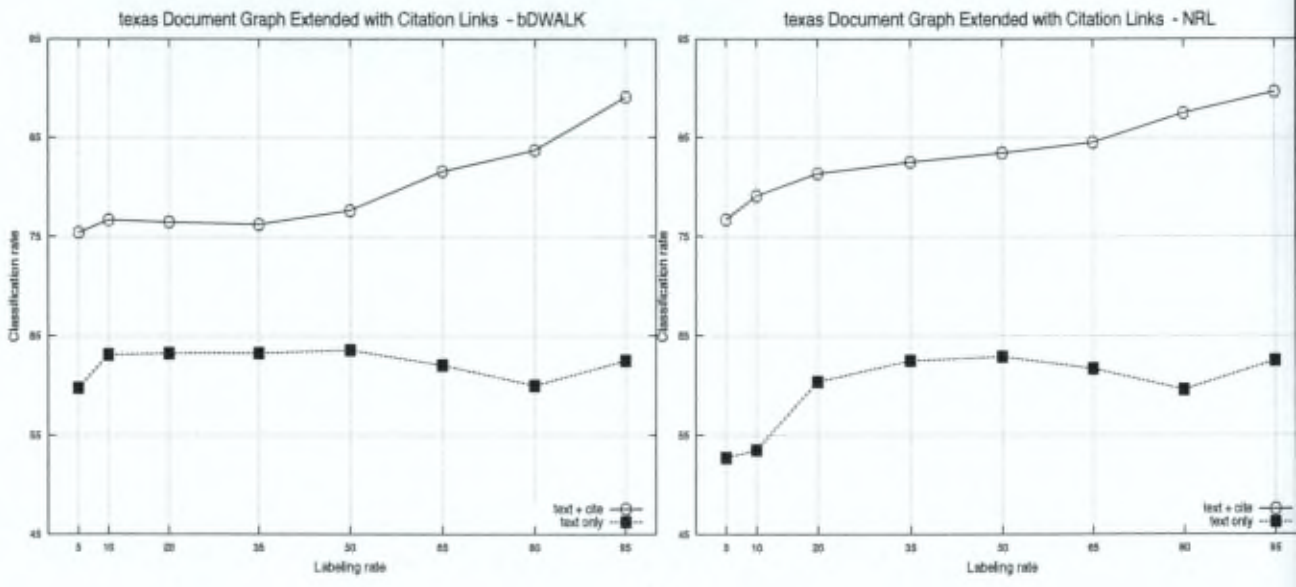
(b)

Figure 7.16: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `U.S. patents` data set.**

Experiment 2 - methodology. As before, the classification accuracy is reported for several labeling rates (5, 10, 20, 35, 50, 65, 80 and 95%), i.e. proportions of nodes for which the label is known. The labels of remaining document nodes are used as test data. For each considered labeling rate, 10 random node label deletions (test sets) were performed (10 runs), on which performances are averaged. For each run, a 10-fold cross-validation is performed on the remaining labeled nodes in order to tune the hyper-parameters of each classifier (see the first experiment in Section 6.3.1 for details), except for the large U.S. patents where the hyper-parameters have been directly tuned with the same values that these used in a previous experiment (Section 6.3.2). The performance on each run is assessed on the remaining unlabeled document nodes (i.e. node that does not appear in the citation graph) with the hyper-parameter tuned during the 10-fold cross-validation. In this second experiment, we report for each data sets the results obtained on two graphs : the weighted 50 nearest neighbors documents graph (text only) and the combined graph. The combined graph is just the union of the real word citation graph and the binary 50 nearest neighbors mined documents graph (text + cite). For memory issues on the `U.S. patents` content-graph, we bound the number of nearest neighbors to 30 .

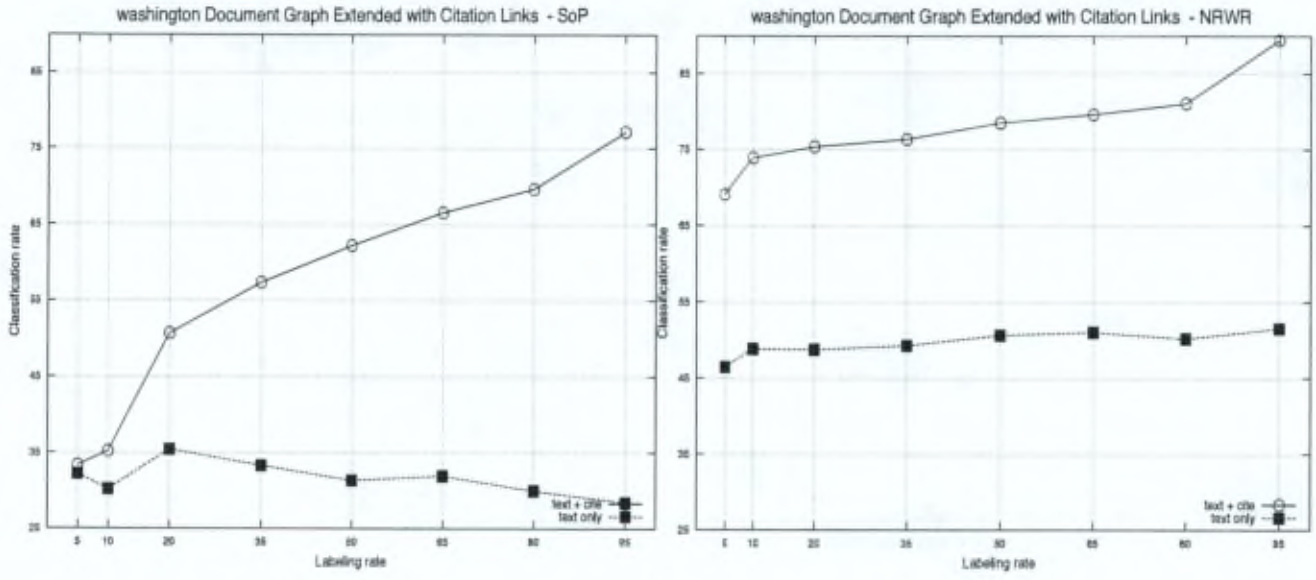


(a)

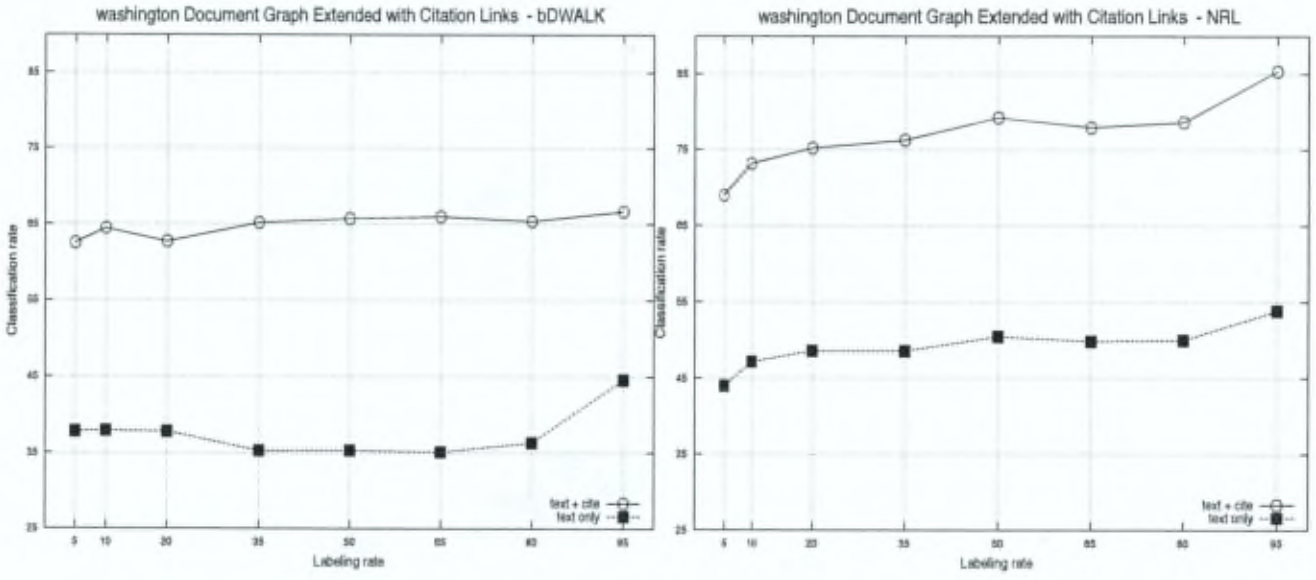


(b)

Figure 7.17: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `WebKB texas` data set.**

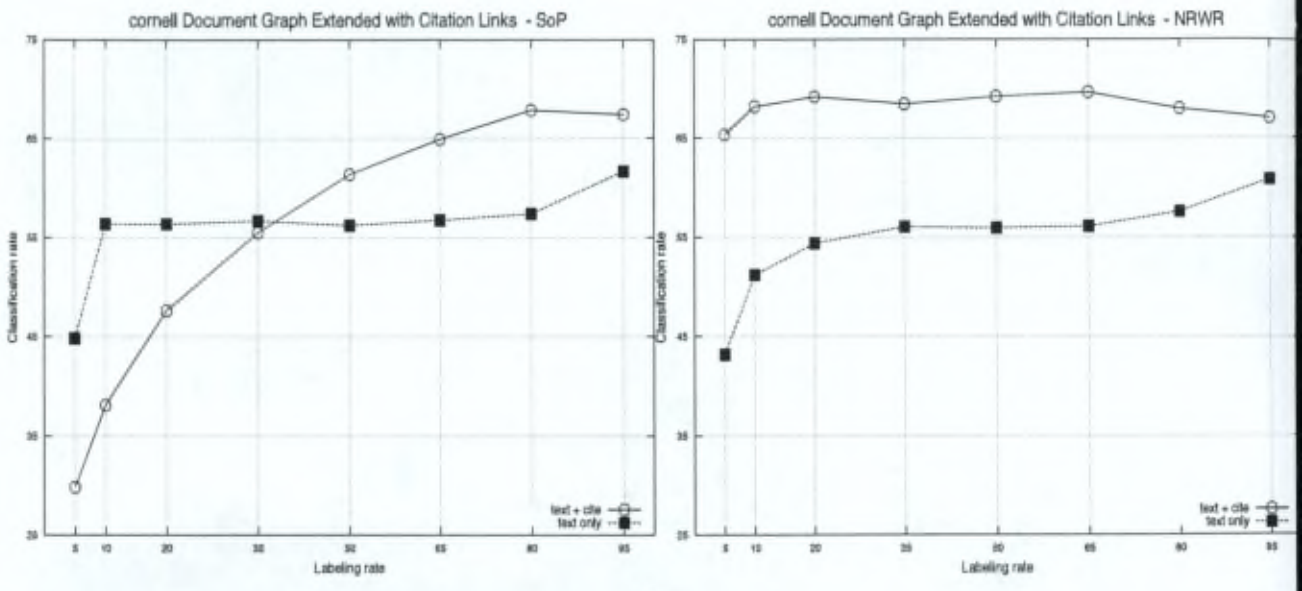


(a)

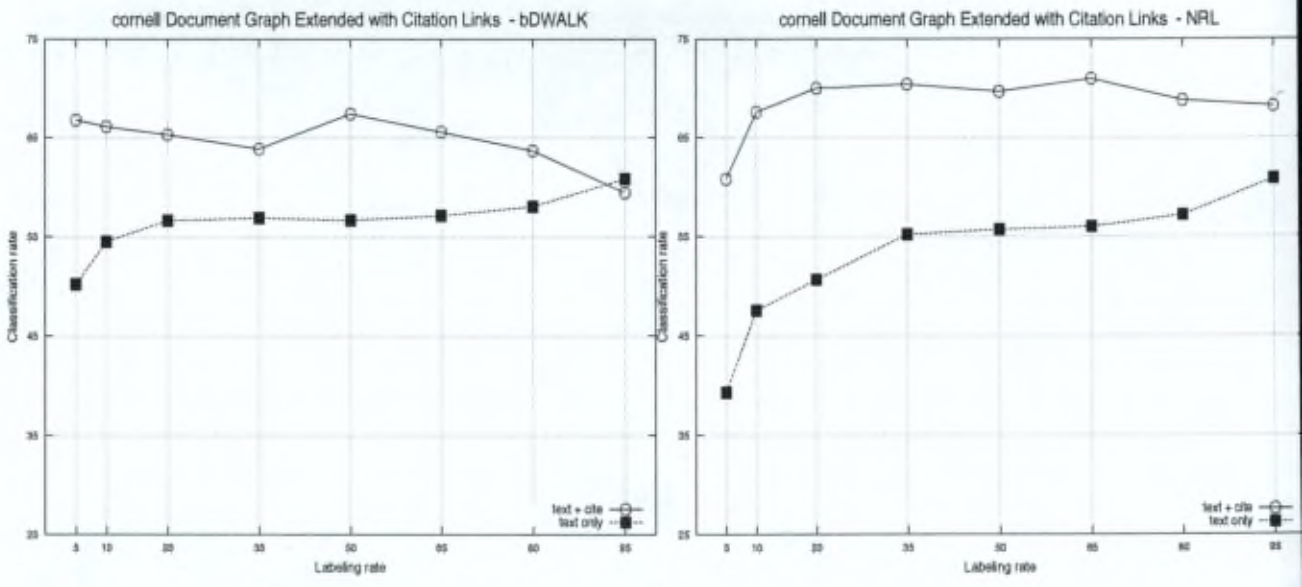


(b)

Figure 7.18: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `WebKB washington` data set.**



(a)



(b)

Figure 7.19: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `WebKB cornell` data set.**

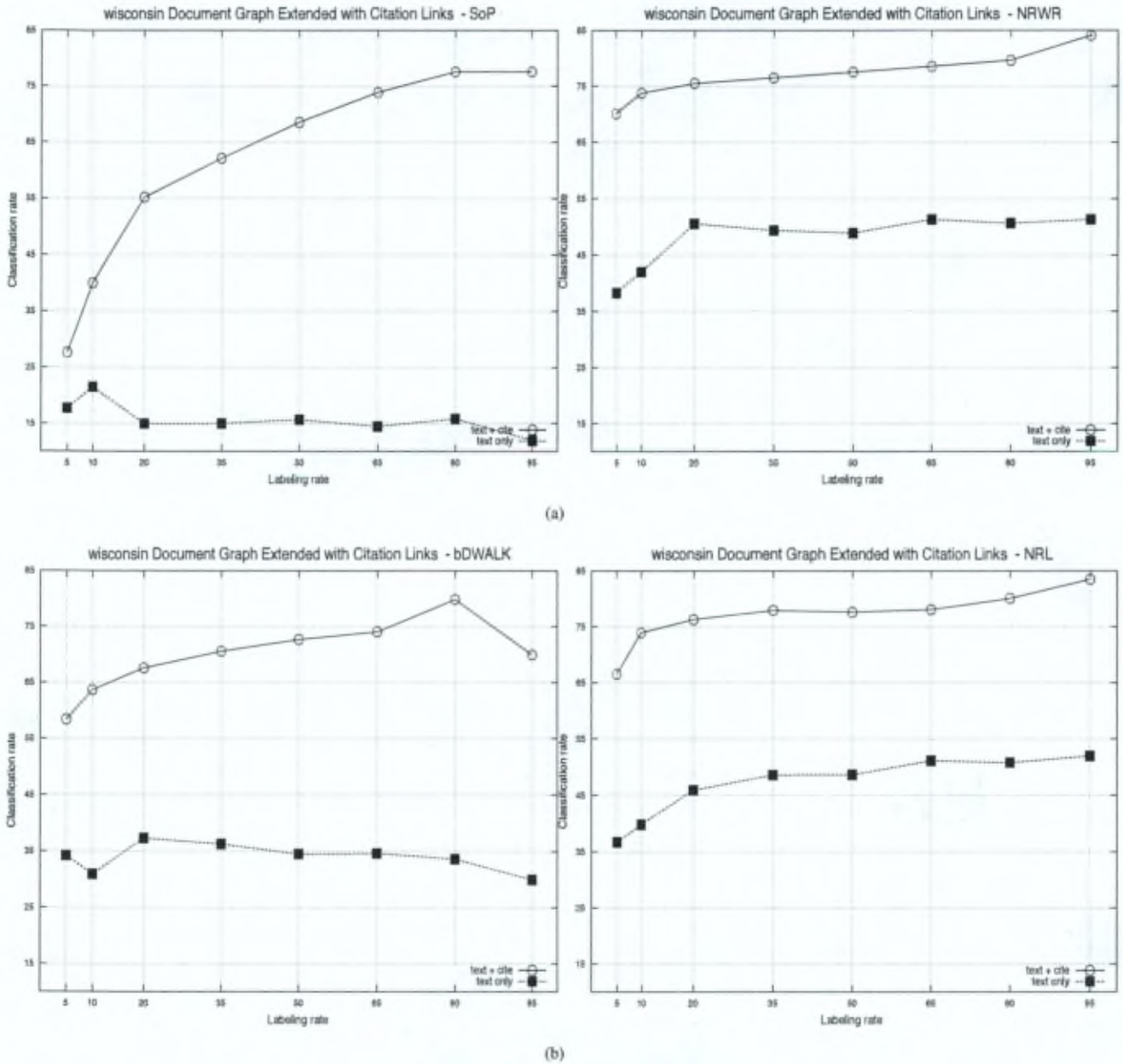


Figure 7.20: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `WebKB wisconsin` data set.**

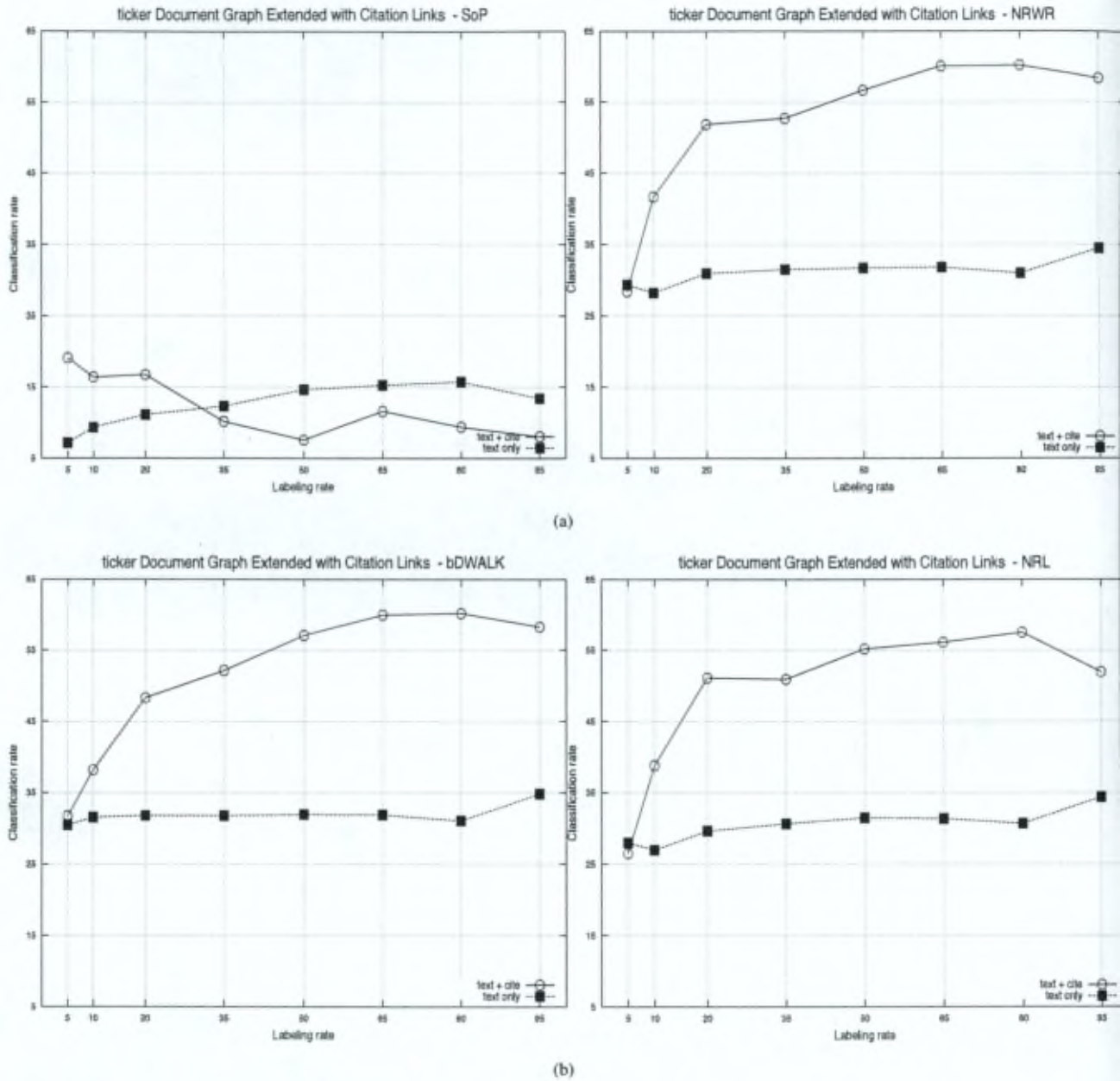
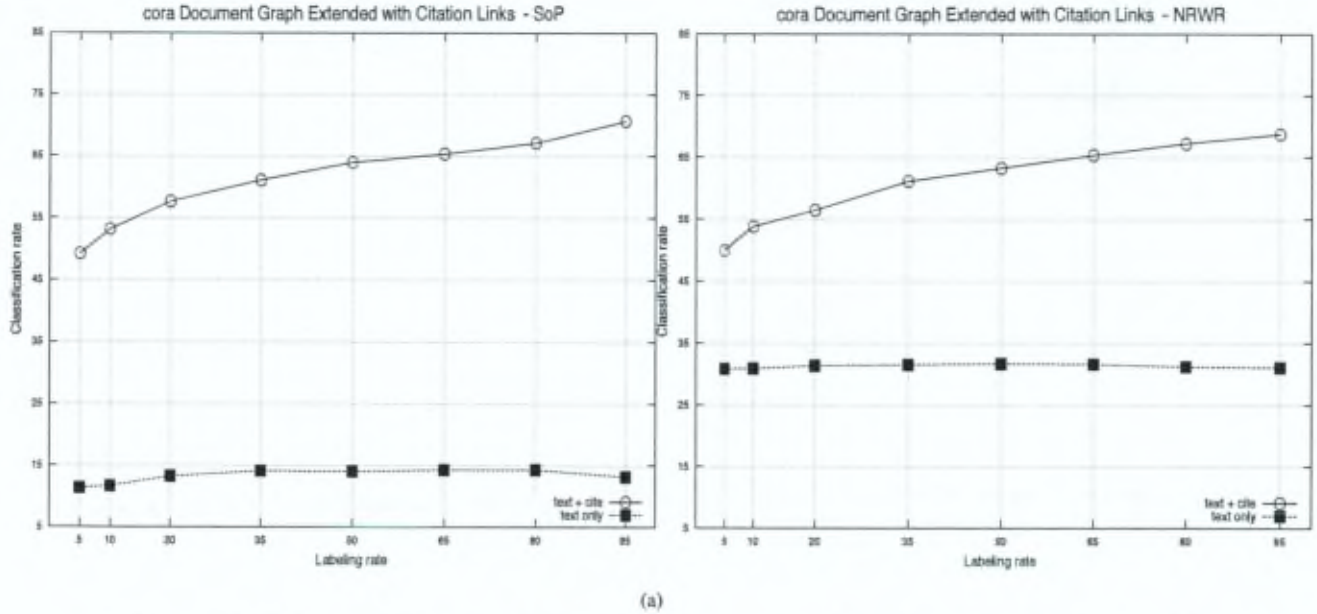
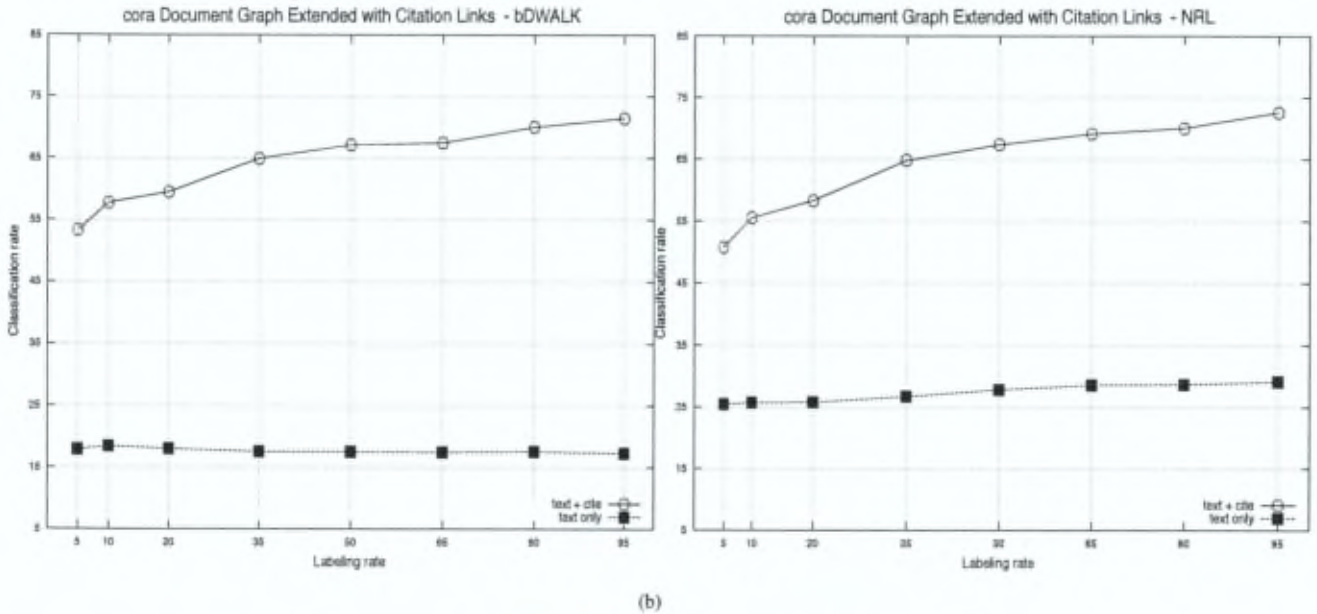


Figure 7.21: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `ticker` data set.**



(a)



(b)

Figure 7.22: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the CoRA data set.**

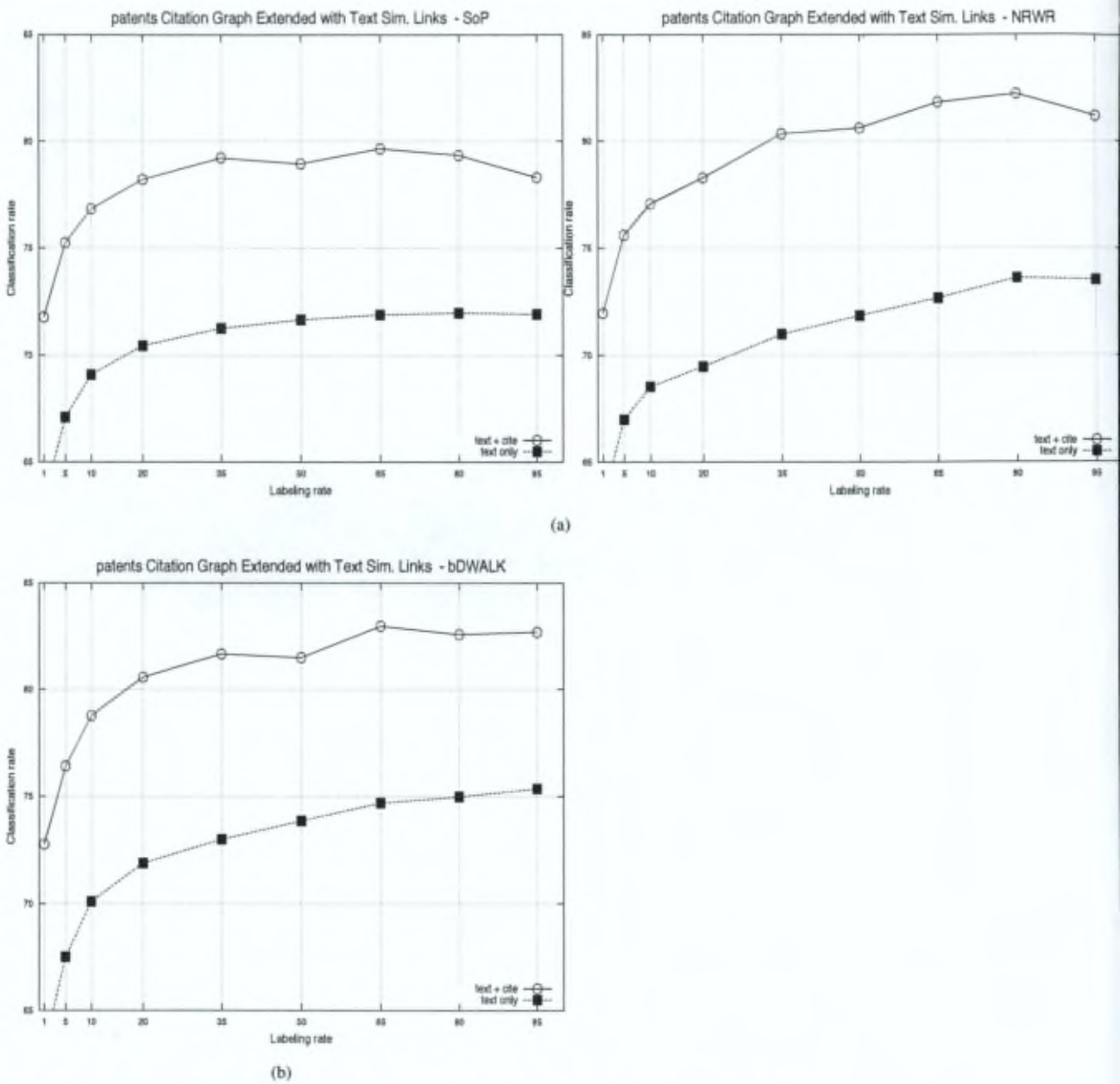


Figure 7.23: **Classification rates in percent, averaged over 10 runs, obtained on partially labeled graphs, for an increasing labeling rate of 5, 10, 20, 35, 50, 65, 80 and 95%. Results are reported for the SoP, NRWR, b$\mathcal{D}$WALK and NRL classification methods. The graphs show the results obtained on the `U.S. patents` data set.**

Results and discussion. Clearly, whatever the data set, the results obtained on the merged graphs (text + cite) outperform those obtained on the mined content textual graphs (see Figures 7.17, 7.18, 7.19, 7.20, 7.21, 7.22 and 7.23). The difference in performance changes according to the dataset, the method and the labeling rate. However, for the three leading methods, NRWR, bDWALK and NRL, using the merged graph (text+cite) always leads to better accuracies. Most of the time, the results on the merged graphs often outperform the inferred content graphs by more than 20% and sometimes more than 30%. This is the case on the `cornell, texas, wisconsin` and `CoRA` data sets. This result is the most important finding of this chapter. Namely, that, when facing a text categorization problem, crawling a citation graph will help to improve drastically classifiers performance. This main result is confirmed on the large scale `U.S. patents` network (Figure 7.23) where the scores are improve up to 10% depending on the algorithm and the labeling rate. In contrast, the SoP sometimes achieves worse results on the combined graph when considering low labeling rates, this is the case on ticker (i.e. industry-pr), cornell and texas data set. Overall, the NRWR and the bDWALK obtain the best results

7.5 Related Work

Enriching existing network with mined links from the available node features in order to improve classification performance is relatively new. For instance, Slattery and Mitchell (2000) exploit hyperlinks between web pages in order to improve traditional classification tasks using only the content (for a good survey on traditional text categorization, see Sebastiani, 2002). Joachims et al. (2001) studied the composition of kernels in order to improve the performance of a soft-margin support vector machine classifier. More precisely, they combined text kernels to co-citation kernels, where a text kernel can be deduced from a bag-of-words representation of the corpus. In such a case, the corpus is represented by a document matrix $\mathbf{D}_{oc}$, whose columns are indexed by the documents and whose rows are indexed by the terms. Therefore, this matrix is also called the *term by document* matrix. The corresponding text kernel is given by the inner product between document vectors (i.e. columns of $\mathbf{D}_{oc}$).

$$\mathbf{K}_{\text{text}}(\mathbf{d}_i, \mathbf{d}_j) = \mathbf{d}_i^{\mathsf{T}} \mathbf{d}_j \tag{7.1}$$

A co-citation matrix is simply a matrix where the two indexed documents (i and j) have a positive score if they are cited by the same document; otherwise the score is set to 0. This matrix is a Gram matrix, since every entry may be seen as the dot product between two document-citation vectors. The feature space associated to this Gram matrix has usually as many dimensions as the number of documents.

In the same spirit, Cohn and Hofmann (2001); Zhu et al. (2007) improved the classifica-

tion performance by using a combination of link-based and content-based probabilistic models. Fisher and Everson (2003) showed that link information can be useful when the document collection has a sufficiently high link density and links are of sufficiently high quality. In the same context, Chakrabarti et al. (1998); Oh et al. (2000) use both local text in a document as well as the distribution of the estimated classes of other documents in its neighborhood, to refine the class distribution of the document being classified. Calado et al. (2003) analyzed several distinct linkage similarity measures and determine which ones provide the best results in predicting the category of a document. They also proposed a Bayesian network model that takes advantage of both the information provided by a content-based classifier and the information provided by the document link structure. Finally, Macskassy (2007) proposes to merge an inferred network and the link network into one global network. Then, he applies to that network the relaxation labeling strategy described in Macskassy and Provost (2007), a baseline algorithm in semi-supervised classification. This algorithm, which is part of the NetKit framework (see Macskassy and Provost, 2007), has been compared to our kernel graph-based algorithms introduced in this thesis (see Section 5.3).

7.6 Conclusions

This chapter investigated how content-based and link-based data sets can be combined for classification purposes. We have empirically shown that citation-based data sets are more valuable compared to content-based data sets for semi-supervised classification, for all investigated data sets. Moreover, combining both sources of information should be considered when facing a text categorization problem. For instance, while classifying journal papers, considering to extract a citation graph might improve significantly the results. In this case, the documents to be classified may benefit from the label propagation coming from the citation graph. However, in another context, when we have a high-quality citation graph, then the inclusion of features on nodes (e.g. text abstracts in the case of papers) does not improve the results, at least for the data sets considered. One interesting perspective will be to investigate other external graph resources. Indeed, up to now, we narrowed the focus of our investigations to citation graphs. For instance, in information retrieval, the use of Wikipedia as external resource constitutes a great challenge for the community (see, e.g., Hu et al., 2009; Lehwark et al., 2010). As future work, we will try to exploit the wikipedia categories with graph-based methods in order to improve semi-supervised classification performances.
Finally, these experiments confirm that the two leading methods are the NRWR and the bDWALK.

Chapter 8

Conclusions and Future Research

8.1 Conclusions

One of the main aspect of graph-based machine learning lies in the design of similarity measures. Indeed, measuring the similarity between pairs of nodes in a graph is one of the key ingredients of graph-based algorithms. There exist plenty of different ways to define such a pairwise measure. Certainly, the most famous one can be derived simply from the shortest-path distance. However, depending on the application and on the field considered, other measures may be more useful. In this thesis, we introduced a novel similarity measure between pairs of nodes of directed graphs: the sum-over-paths covariance. This measure has an intuitive and precise interpretation: two nodes are highly correlated if they often co-occur together in the same, preferably short paths. This measure can be tuned through a temperature parameter in order to bias walks towards or against shortest-paths. This way, we were able to obtain a nice framework that can be used in many different applications where the shortest-path is not necessarily the best choice. For instance, in the case of HMMs (Hidden Markov Models) two famous algorithms are commonly used for training. First, in the Baum-Welch algorithm parameters are assessed on all possible paths that match the inputs. Second, in the Viterbi training algorithm parameters are estimated only on the best paths (i.e. the Viterbi paths). Our novel measure has a simple quantitative parameter that can be tuned in order to cover the complete spectrum going from a natural random walk (in which all paths are equiprobable) to walks that are biased towards the shortest-paths. This measure, constitutes the first main contribution of this work. It can be computed easily and has the complexity of a matrix inversion.

This leads to the second main contribution of the thesis: How to use such measures on

large scale networks? Indeed, a matrix inversion can not be computed if the number of rows (i.e. of nodes) is too large (> 50,000). However, depending on the application, we do not really need to compute explicitly all pairwise similarities between the nodes of a graph. In this thesis, we narrowed the focus on the case of semi-supervised learning which received a growing interest in the last decade. In this case, based on a well known regularization framework introduced by Zhou et al. (2003, 2005), we computed a sum-of-similarities which corresponds to a product $\mathbf{Ky}$, where $\mathbf{K}$ is the similarity measure matrix associated with the graph and $\mathbf{y}$ is a binary class indicator column vector. We showed how it is possible to compute this sum-of-similarities directly for the sum-over-paths covariance measure and for the normalized random walk with restart measure (NRWR). Another significant contribution, is to propose a generalization of the $\mathcal{D}$WALK algorithm based on the randomized shortest path framework of Saerens et al. (2009). All these algorithms have a linear computing time in the number of edges, classes and steps, and hence can be applied to large scale networks. The algorithms were benchmarked on medium-size data sets, and obtained competitive results on the `U.S. patents` citation network. Moreover, it was shown that the b$\mathcal{D}$WALK and the approximate NRWR are very competitive in case of low labeling rates, even better than state-of-the-art methods. Both algorithms require just a few minutes to classify millions of nodes (see Table 6.5). Furthermore, during the thesis, we collected a novel benchmark data set: the `U.S. patents` citation network. This data set is now available to the community for benchmarks purposes.

The last part of the thesis investigated how to combine a citation graph to information on its nodes. We first showed that citation-based data sets are more accurate for classification than content-based data sets. Indeed, in our experiments, on seven different data sets, we observed a difference between 10 to 30% of the classification rate in favor of citation graphs. This fact lead us to investigate the exploitation of citation links in order to improve text categorization. This was achieved by connecting documents to an external citation graph on the basis of textual similarities. On the merged graph (i.e. citation + content) we were able to significantly improve the classification rate of the documents that do not appear in the external citation graph. Indeed, we observed an upgrade from 10 up to 30% in classification accuracy. In contrast, while working on this same merged graph, we were not able to improve the classification rate of the citation nodes. All these results ensue from the main observation that citation-based graphs seem to be more accurate for classification than content-based graphs. In this final part, again, the b$\mathcal{D}$WALK and the approximate NRWR are very competitive on medium-size and large-scale networks (see see Section 7.3 and Section 7.2). Both are more robust than other methods to the variations of the number of neighbors retained to build the graph (see Section 7.3). Moreover, they obtain competitive results in comparison with a linear support vector machine based on the content only.

8.2 Perspectives

We present here some perspectives that look interesting.

In this thesis we speak about semi-supervised classification, but the scope has been restricted to transductive methods. In other words, our algorithms can only predict labels for test points that are part of the model. When having a new point, a transductive method has to relaunch all the classification procedure ignoring previous results. One interesting perspective would be to try to infer inductive models that can classify new points using what was previously learned. One possibility to follow up may lie on the use of the Sherma-Morrison formula to compute a new kernel integrating new points based on a previously computed kernel. Deriving a new inductive classification procedure is a challenging perspective.

Another not related perspective appears, for example, when trying to apply our classification methods on large scale networks. Two global approaches have been introduced in this thesis in order to decrease the complexity of the algorithms. One of them consists in solving linear systems up to convergence. This technique although efficient may break the intuitive interpretation of the measure we are approximating. This is the case for the sum-over-paths covariance. Indeed, in this case, by approximating the classification scores by the convergence of linear systems, the results can no more be intuitively interpreted in terms of walks in a graph. In order to keep the underlying interpretation, it would better to directly bound the measure up to a specific walk length τ. This can be done, as for the biased $\mathcal{D}$WALK, by working directly on a lattice L derived from the original network. So, a further work consists in deriving a classification procedure based on the sum-over-path covariance using a forward/backward algorithm.

In another context, the b$\mathcal{D}$WALK defines walks in a network starting and ending in the same class. Hence, we obtain a per-class betweenness. One interesting issue, would be also to investigate the case of starting and ending in different classes. Such a setting may be interesting for classification where we would operate in a one against one basis.

Up to now, we considered only standard graph structures with binary relations between nodes. However in current large-scale data sets, we observe ternary relations and more. For example, on youtube a video is associated to a set of users and also to a set of comments. In this context, studying the generalization of graphs to hyper-graphs constitutes an important step. More precisely, as future work, we want to generalize our three proposed algorithms to the case of hyper-graphs. This leads to interesting theoretical and applied perspectives.

Another application would be the use of Wikipedia as external resource, which constitutes a hot topic in the community at the moment (see, e.g., Hu et al., 2009; Lehwark et al., 2010). Wikipedia forms a wide partially labeled (wikipedia categories) network and hence is an ideal candidate for our algorithms. As future work, we will try to exploit the wikipedia categories with graph-based methods in order to improve semi-

supervised classification performance.

Finally, another more practical perspective would be to apply our algorithms on patent data to measure technological innovation. Patent data analysis has been a common practice in economics for the past decades. It typically consists in simple counts, sometimes weighted by different value indicators (Griliches, 1990; Greenhalgh and Rogers, 2006). At best, these weighting schemes are made of counts of citations received from subsequent patents. Such indicators make therefore a very limited use of the richness of the citations graph and no use whatsoever of the textual contents of the patents, although it is precisely what determines the legal and technological scope of an invention. The patent based indicators that are currently used by economists are therefore unsatisfactory in their ability to inform research on the economics of innovation and particularly on the value of technological inventions and on the effect of patents on competitive conditions. Graph mining and link analysis algorithms applied to patent databases have the potential to improve our understanding of these phenomena in different ways. First, measures of prestige and centrality (such as the PageRank and HITS algorithms) can improve raw citation counts by accounting for the recursive nature of citations, thereby improving our measures of the value of inventions. Second, similarity measures and classification algorithms — particularly those that can leverage both the citations graph and the textual contents — are key to detect multiple patents covering a single invention, which is key to understand the fragmentation of intellectual property rights, and could greatly improve the identification of technological areas beyond the static patent classifications such as the IPC. This is essential in an era in which technologies have a greater propensity to cross-pollinate over different fields, and research becomes increasingly multidisciplinary, making it ever harder to delineate different technology fields. We think that, by dynamically adapting to emerging trends in patenting and citation patterns, graph- and text-based classification algorithms could therefore improve the measurement of scientific and technological development. Finally, measures of graph-density or crowdedness should enable us to identify so-called "patent thickets", these dense webs of patents the ownership of which is greatly fragmented between different firms. Economists increasingly fear that such thickets hamper competition and innovation, but can hardly observe such cases to analyze them more carefully (Hall, 2001; Bessen, 2003; Harhoff et al., 2007). Overall, these different algorithms should also improve the detection and quantification of knowledge flows across firms, industries or geographic areas (Jaffe et al., 1993; Cassiman and Veugelers, 2002). Hence, using the algorithms and ideas introduced in this thesis for the analysis of patents may bring more satisfactory results for economists and constitutes an important challenge.

Appendix A

Computation of the partial derivatives of the partition function $\mathcal{Z}$

First, let us compute the expected energy $\overline{E}$ given by Equation (4.14)

$$\begin{aligned} \overline{E} &= \frac{\partial(-\ln \mathcal{Z})}{\partial\theta} = -\frac{\partial_\theta \left[\mathbf{e}^{\mathsf{T}}((\mathbf{I}-\mathbf{W})^{-1}-\mathbf{I})\mathbf{e}\right]}{\mathcal{Z}} \\ &= -\frac{\mathbf{e}^{\mathsf{T}}\partial_\theta(\mathbf{I}-\mathbf{W})^{-1}\mathbf{e}}{\mathcal{Z}} \end{aligned} \tag{A.1}$$

where the partition function $\mathcal{Z}$ is provided by Equation (3.13). We thus have to compute $\partial_\theta(\mathbf{I}-\mathbf{W})^{-1}$; by setting $\mathbf{Z} = (\mathbf{I}-\mathbf{W})^{-1}$ and denoting element k, k' of $\mathbf{Z}$ by $z_{kk'}$, we obtain

$$\begin{aligned} \partial_\theta(\mathbf{I}-\mathbf{W})^{-1} &= -\mathbf{Z}(\partial_\theta(\mathbf{I}-\mathbf{W}))\mathbf{Z} \\ &= \mathbf{Z}(\partial_\theta\mathbf{W})\mathbf{Z} \\ &= \mathbf{Z}\mathbf{W}'_\theta\mathbf{Z} \end{aligned} \tag{A.2}$$

where the matrix $\mathbf{W}'_\theta$ contains the elements $[\mathbf{W}'_\theta]_{kk'} = -c_{kk'}\exp\left[-\theta c_{kk'} + \ln p^{\text{ref}}_{kk'}\right]$. In matrix form, $\mathbf{W}'_\theta = -\mathbf{C}\circ\mathbf{W}$ where $\circ$ is the elementwise (Hadamard) matrix product Harville (1997). Therefore, the expected energy, $\overline{E}$, is

$$\overline{E} = -\frac{\mathbf{e}^{\mathsf{T}}\mathbf{Z}\mathbf{W}'_\theta\mathbf{Z}\mathbf{e}}{\mathcal{Z}} \tag{A.3}$$

We now turn to the computation of $\overline{\eta}(k,k')$ (Equation (4.17)). Recall that F is the free energy defined in Equation (4.13).

$$\overline{\eta}(k,k') = \frac{\partial F}{\partial c_{kk'}} \tag{A.4}$$

$$= -\frac{1}{\theta}\frac{\partial_{c_{kk'}}\left[\mathbf{e}^{\mathrm{T}}((\mathbf{I}-\mathbf{W})^{-1}-\mathbf{I})\mathbf{e}\right]}{\mathcal{Z}} \tag{A.5}$$

$$= -\frac{1}{\theta}\frac{\mathbf{e}^{\mathrm{T}}\partial_{c_{kk'}}(\mathbf{I}-\mathbf{W})^{-1}\mathbf{e}}{\mathcal{Z}} \tag{A.6}$$

$$= -\frac{1}{\theta}\frac{\mathbf{e}^{\mathrm{T}}(\partial_{c_{kk'}}\mathbf{Z})\mathbf{e}}{\mathcal{Z}} \tag{A.7}$$

Let us compute $\partial_{c_{kk'}}\mathbf{Z}$,

$$\partial_{c_{kk'}}\mathbf{Z} = \partial_{c_{kk'}}(\mathbf{I}-\mathbf{W})^{-1} \tag{A.8}$$

$$= -\mathbf{Z}(\partial_{c_{kk'}}(\mathbf{I}-\mathbf{W}))\mathbf{Z} \tag{A.9}$$

$$= \mathbf{Z}(\partial_{c_{kk'}}\mathbf{W})\mathbf{Z} \tag{A.10}$$

$$= -\theta\,\exp\left[-\theta c_{kk'}+\ln p^{\mathrm{ref}}_{kk'}\right]\,\mathbf{Z}\mathbf{e}_k\mathbf{e}_{k'}^{\mathrm{T}}\mathbf{Z} \tag{A.11}$$

Thus, by defining $z_{\bullet k}=\sum_{i=1}^{n} z_{ik}$ and $z_{k\bullet}=\sum_{i=1}^{n} z_{ki}$, $\overline{\eta}(k,k')$ is given by

$$\overline{\eta}(k,k') = \exp\left[-\theta c_{kk'}+\ln p^{\mathrm{ref}}_{kk'}\right]\frac{\mathbf{e}^{\mathrm{T}}\mathbf{Z}\mathbf{e}_k\mathbf{e}_{k'}^{\mathrm{T}}\mathbf{Z}\mathbf{e}}{\mathcal{Z}} \tag{A.12}$$

$$= \frac{\sum_{i,j=1}^{n} z_{ik}z_{k'j}\exp\left[-\theta c_{kk'}+\ln p^{\mathrm{ref}}_{kk'}\right]}{\mathcal{Z}} \tag{A.13}$$

$$= \frac{z_{\bullet k}z_{k'\bullet}\exp\left[-\theta c_{kk'}+\ln p^{\mathrm{ref}}_{kk'}\right]}{\mathcal{Z}} \tag{A.14}$$

The expected number of passages through node k', that is, the betweenness measure (Equation (4.20)), is

$$\mathrm{bet}(k') = \sum_{k=1}^{n}\overline{\eta}(k,k') \tag{A.15}$$

$$= \frac{\sum_{k=1}^{n} z_{\bullet k}z_{k'\bullet}\exp\left[-\theta c_{kk'}+\ln p^{\mathrm{ref}}_{kk'}\right]}{\mathcal{Z}} \tag{A.16}$$

$$= \frac{(z_{\bullet k'}-1)z_{k'\bullet}}{\mathcal{Z}} \tag{A.17}$$

where we used Equation (4.31). The second-order derivative (Equation (4.21)) is a bit tedious to compute; it aims to differentiate $\overline{\eta}(k,k')$ provided by Equation (A.12):

$$\overline{\eta}(k,k';l,l') = \frac{1}{\theta^2}\frac{\partial^2(\ln \mathcal{Z})}{\partial c_{ll'}\partial c_{kk'}} \tag{A.18}$$

$$= -\frac{1}{\theta}\frac{\partial(\overline{\eta}(k,k'))}{\partial c_{ll'}} \tag{A.19}$$

$$\begin{aligned} = & -\frac{\exp\left[-\theta c_{kk'} + \ln p^{\text{ref}}_{kk'}\right]}{\theta}\left\{-\theta\frac{\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}}{\mathcal{Z}}\delta_{kl}\delta_{k'l'}\right. \\ & +\frac{\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}}{\mathcal{Z}}(\partial_{c_{ll'}}\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k) + \frac{\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k}{\mathcal{Z}}(\partial_{c_{ll'}}\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}) \\ & \left.-\frac{\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}}{\mathcal{Z}^2}(\partial_{c_{ll'}}\mathcal{Z})\right\} \end{aligned} \tag{A.20}$$

$$\begin{aligned} = & \exp\left[-\theta c_{kk'} + \ln p^{\text{ref}}_{kk'}\right]\left\{\frac{\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}}{\mathcal{Z}}\delta_{kl}\delta_{k'l'}\right. \\ & +\exp\left[-\theta c_{ll'} + \ln p^{\text{ref}}_{ll'}\right]\left[\frac{\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}}{\mathcal{Z}}(\mathbf{e}^\top\mathbf{Z}\mathbf{e}_l\mathbf{e}_{l'}^\top\mathbf{Z}\mathbf{e}_k)\right. \\ & +\frac{\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k}{\mathcal{Z}}(\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}_l\mathbf{e}_{l'}^\top\mathbf{Z}\mathbf{e}) \\ & \left.\left.-\frac{\mathbf{e}^\top\mathbf{Z}\mathbf{e}_k\mathbf{e}_{k'}^\top\mathbf{Z}\mathbf{e}}{\mathcal{Z}^2}\mathbf{e}^\top(\mathbf{Z}\mathbf{e}_l\mathbf{e}_{l'}^\top\mathbf{Z})\mathbf{e}\right]\right\} \end{aligned} \tag{A.21}$$

$$\begin{aligned} = & \exp\left[-\theta c_{kk'} + \ln p^{\text{ref}}_{kk'}\right]\left\{\frac{z_{\bullet k}z_{k'\bullet}}{\mathcal{Z}}\delta_{kl}\delta_{k'l'}\right. \\ & +\exp\left[-\theta c_{ll'} + \ln p^{\text{ref}}_{ll'}\right]\left[\frac{z_{k'\bullet}z_{\bullet l}z_{l'k}}{\mathcal{Z}}\right. \\ & \left.\left.+\frac{z_{l'\bullet}z_{\bullet k}z_{k'l}}{\mathcal{Z}} - \frac{z_{k'\bullet}z_{\bullet k}z_{l'\bullet}z_{\bullet l}}{\mathcal{Z}^2}\right]\right\} \end{aligned} \tag{A.22}$$

Finally, using Equations (4.29-4.32), the corresponding covariances between nodes (Equation (4.25)) are

$$\text{cov}(k',l') = \sum_{k,l=1}^{n}\overline{\eta}(k,k';l,l') \tag{A.23}$$

$$
\begin{aligned}
&= \sum_{k=1}^{n} \exp\left[-\theta c_{kk'} + \ln p_{kk'}^{\text{ref}}\right] \\
&\quad \times \left\{ \frac{z_{\bullet k} z_{k'\bullet}}{\mathcal{Z}} \delta_{k'l'} + \frac{z_{k'\bullet}(z_{\bullet l'} - 1) z_{l'k}}{\mathcal{Z}} \right. \\
&\quad \left. + \frac{z_{l'\bullet} z_{\bullet k}(z_{k'l'} - \delta_{k'l'})}{\mathcal{Z}} - \frac{z_{k'\bullet} z_{\bullet k} z_{l'\bullet}(z_{\bullet l'} - 1)}{\mathcal{Z}^2} \right\} \qquad \text{(A.24)} \\
&= \frac{(z_{\bullet k'} - 1) z_{k'\bullet}}{\mathcal{Z}} \delta_{k'l'} + \frac{z_{k'\bullet}(z_{\bullet l'} - 1)(z_{l'k'} - \delta_{l'k'})}{\mathcal{Z}} \\
&\quad + \frac{z_{l'\bullet}(z_{\bullet k'} - 1)(z_{k'l'} - \delta_{k'l'})}{\mathcal{Z}} \\
&\quad - \frac{z_{k'\bullet}(z_{\bullet k'} - 1) z_{l'\bullet}(z_{\bullet l'} - 1)}{\mathcal{Z}^2} \qquad \text{(A.25)} \\
&= \frac{1}{\mathcal{Z}} \left\{ (z_{\bullet k'} - 1) z_{k'\bullet} \delta_{k'l'} + z_{k'\bullet}(z_{\bullet l'} - 1)(z_{l'k'} - \delta_{l'k'}) \right. \\
&\quad + z_{l'\bullet}(z_{\bullet k'} - 1)(z_{k'l'} - \delta_{k'l'}) \\
&\quad \left. - \frac{z_{k'\bullet}(z_{\bullet k'} - 1) z_{l'\bullet}(z_{\bullet l'} - 1)}{\mathcal{Z}} \right\} \qquad \text{(A.26)}
\end{aligned}
$$

Appendix B

List of Figures

Appendix C

List of Tables

Acronyms

SoP	sum-over-paths
aSoP	approximated sum-over-paths
NRL	normalized regularized laplacian
aNRL	approximated normalized regularized laplacian
NRWR or RCT	normalized random walk with restart or regularized commute-time
bNRWR	bounded normalized random walk with restart
bDWALK	biased discriminative walk
hit Time	hitting time
KNN	k nearest neighbors
CT	commute-time
NCT	normalized commute-time
DM	diffusion map
RL	regularized laplacian
RSP	randomized shortest path
SVM	support vector machine

Notations

$\mathcal{G}$	a weighted graph not necessarily connected
V	The set of nodes or vertices of $\mathcal{G}$
E	The set of arcs or edges of $\mathcal{G}$
n	The number of nodes of the graph $\mathcal{G}$
$\mathbf{A}$	The adjacency matrix of $\mathcal{G}$
a_{ij}, $[\mathbf{A}]_{ij}$	The element in the ith row and the jth column of the matrix $\mathbf{A}$
$a_{i\bullet}$	The sum of the elements of the ith row of the matrix $\mathbf{A}$
$a_{\bullet i}$	The sum of the elements of the ith column of the matrix $\mathbf{A}$
$a_{\bullet\bullet}$	The sum all the elements of the matrix $\mathbf{A}$
$\mathbf{v}$	a column vector
$\mathbf{v}^{\mathrm{T}}$	a row vector
v_i, $[\mathbf{v}]_i$	ith element of vector $\mathbf{v}$
$\mathbf{D}$	The outdegree matrix of the graph $\mathcal{G}$
$\mathbf{C}$	The cost matrix associated to $\mathbf{A}$
$\mathbf{W}$	The weight matrix associated to $\mathbf{A}$
$\mathbf{P}$	The transition probabilities matrix of the graph $\mathcal{G}$
$\mathbf{L}$	The laplacian matrix of the graph
$\mathbf{I}$	The identity matrix
$\mathbf{e}$	a unit column vector full of 1's
$\mathbf{0}$	a null column vector full of 0's
$\mathbf{e}_i$	ith of column $\mathbf{I}$
$\mathbf{K}$	a kernel matrix
θ	temperature parameter
$\overline{\eta}(k,k')$	The expected number of transitions through the link $k \to k'$
$\mathrm{bet}(k) = \overline{n}_k$	the expected number of passages in node k, the betweenness of node k
$\mathrm{cov}(k,l)$	the covariance between node k and node l
$\mathrm{cor}(k,l)$	the correlation between node k and node l
δ_{kl}	the Kronecker delta whose value is 1 if $k = l$ and 0 otherwise
L	the lattice build from the original network $\mathcal{G}$
τ	the maximum walk length, or the maximum number of steps of an iterative procedure
$\mathcal{L}$	the set of labels

Bibliography

A

R. Agaev and P. Chebotarev. The matrix of maximum out forests of a digraph and its applications. *Automation and Remote Control*, 61(9):1424–1450, 2000. [p. 37]

R. Agaev and P. Chebotarev. Spanning forests of a digraph and their applications. *Automation and Remote Control*, 62(3):443–466, 2001. [p. 37]

T. Akamatsu. Cyclic flows, markov process and stochastic traffic assignment. *Transportation Research B*, 30(5):369–386, 1996. [p. 18 and 22]

B

F. Bach and M.I. Jordan. Predictive low-rank decomposition for kernel methods. *Proceedings of the Twenty-second International Conference on Machine Learning*, 22, 2005. [p. 67]

Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*. Addison-Wesley, 1999. [p. 14]

R. B. Bapat. Resistance distance in graphs. *The Mathematics Student*, 68:87–98, 1999. [p. 36]

A. Bavelas. Communication patterns is task-oriented groups. *Journal of the Acoustical Society of America*, 22:271–282, 1950. [p. 26]

M. Belkin, I. Matveeva, and P. Niyogi. Tikhonov regularization and semi-supervised learning on large graphs. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP2004)*, pages 1000–1003, 2004a. [p. 51, 60, 61, 66, 69, and 90]

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from examples. *Technical Report TR-2004-06*, 2004b. [p. 55, 60, 61, and 66]

M. Belkin, P. Niyogi, and V. Sindhwani. On manifold regularization. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 17 – 24, 2005. [p. 55, 61, and 66]

Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1): 87–90, 1958. [p. 17]

Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2000. [p. 17]

J.E. Bessen. Patent thickets: Strategic patenting of complex technologies. 2003. [p. 132]

Chawla S. Blum A. Learning from labeled and unlabeled data using graph mincuts. In *In the proceedings of the International Conference of Machine Learning*, 2001. [p. 90]

Bela Bollobas. *Modern graph theory*. Springer, 1998. [p. 7]

I. Borg and P. Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 1997. [p. 13, 15, and 58]

M. Brand. A random walks perspective on maximizing satisfaction and profit. *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005. [p. 33, 74, and 90]

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998. [p. 25, 33, 36, and 55]

Richard Bronson. *Matrix operations*. McGraw-Hill, 1989. [p. 23]

C

P. Calado, M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M.A. Gonçalves. Combining link-based and content-based methods for web document classification. In *Proceedings of the twelfth international conference on Information and knowledge management*, page 401. ACM, 2003. [p. 128]

J. Callut, K. Francoisse, M. Saerens, and P. Dupont. Semi-supervised classification from discriminative random walks. In *Proceedings of the European conference on Machine Learning (ECML 2008)*, volume LNAI5211, pages 162–177, 2008. [p. 2, 46, 68, 69, 73, 75, 90, and 91]

B. Cassiman and R. Veugelers. R&D cooperation and spillovers: some empirical evidence. *The American Economic Review*, 92, 2002. [p. 86 and 132]

Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998. [p. 66 and 128]

T.F. Chan, P. Ciarlet, and W.K. Szeto. On the optimality of the median cut spectral bisection graph partitioning method. *SIAM Journal on Scientific Computing*, 18(3): 943–948, 1997. [p. 33]

A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *Annual ACM Symposium on Theory of Computing*, pages 574–586, 1989. [p. 31, 32, and 36]

O. Chapelle, J. Westion, and B. Scholkopf. Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, 2002. [p. 90]

O. Chapelle, B. Scholkopf, and A. Zien. *Semi-supervised learning*. The MIT Press, 2006. [p. 66, 68, 69, 83, 87, and 97]

P. Chebotarev. A new family of graph distances. *Available at arxiv.org as manuscript arXiv:0810.2717v2*, 2008. [p. 33]

P. Chebotarev and E. Shamis. The matrix-forest theorem and measuring relations in small social groups. *Automation and Remote Control*, 58(9):1505–1514, 1997. [p. 30, 31, and 58]

P. Chebotarev and E. Shamis. On proximity measures for graph vertices. *Automation and Remote Control*, 59(10):1443–1459, 1998. [p. 30, 31, and 58]

P. Chebotarev and E. Shamis. The forest metric for graph vertices. *Electronic Notes in Discrete Mathematics*, 11:98–107, 2002. [p. 67]

Dechang Chen and Xiuzhen Cheng. An asymptotic analysis of some expert fusion methods. *Pattern Recognition Letters*, 22:901–904, 2001. [p. 109]

J. Chen and J. Ye. Training svm with indefinite kernels. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 136–143, 2008. [p. 13]

Tsinghua Chen, Qiong Yang, and Xiaoou Tang. Directed graph embedding. *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2707–2712, 2007. [p. 37]

Y. Chen, E. Garcia, M. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: concepts and algorithms. *Journal of Machine Learning Research*, 10:747–776, 2009. [p. 13]

F. R. Chung. *Spectral graph theory*. American Mathematical Society, 1997. [p. 7, 29, and 32]

F. R. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9:1–19, 2005. [p. 37]

D. Cohn and T. Hofmann. The missing link-a probabilistic model of document content and hypertext connectivity. 2001. [p. 127]

Roger M. Cooke. *Experts in uncertainty*. Oxford University Press, 1991. [p. 109]

Adrian Corduneanu. *The Information Regularization Framework for Semi-supervised Learning*. PhD thesis, Massachusetts Institute of Technology, 2006. [p. 66]

R. Cowan and D. Foray. The economics of codification and the diffusion of knowledge. *Industrial and Corporate Change*, 6, 1997. [p. 85]

R. Cowan and D. Foray. The explicit economics of knowledge codification and tacitness. *Industrial and Corporate Change*, 9, 2000. [p. 86]

T. Cox and M. Cox. *Multidimensional scaling, 2nd ed.* Chapman and Hall, 2001. [p. 13 and 15]

D

Timothy A. Davis. *Direct methods for sparse linear systems*. SIAM, 2006. [p. 45]

P. De Wagter. Combining graph topology with node features for semi supervised classification, 2010. [p. 97]

Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. [p. 17]

L. Donetti and M. Munoz. Detecting network communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, P10012, 2004. [p. 33]

Peter G. Doyle and J. Laurie Snell. *Random Walks and Electric Networks*. The Mathematical Association of America, 1984. [p. 8, 13, 23, and 36]

Didier Dubois, Michel Grabisch, Henri Prade, and Philippe Smets. Assessing the value of a candidate: Comparing belief function and possibility theories. *Proceedings of the Fifteenth international conference on Uncertainty in Artificial Intelligence*, pages 170–177, 1999. [p. 109]

P. Dupont. *Probabilistic sequence modeling: An introduction to Markov Chains and Hidden Markov Models*, may 2007. Doctoral course CIL. [p. 13]

F

M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619–633, 1975. [p. 33]

S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001. [p. 46]

M. Fisher and R. Everson. When Are Links Useful? Experiments in Text Classification. In *Advances in information retrieval: 25th European Conference on IR Research*, page 41. Springer Verlag, 2003. [p. 128]

L. R. Ford and D. R. Fulkerson. Flows in networks. *Princeton University Press*, 1962. [p. 17]

F. Fouss, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of graph kernels on a collaborative recommendation task. *Proceedings of the 6th International Conference on Data Mining (ICDM 2006)*, pages 863–868, 2006. [p. 37, 58, and 67]

F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph, with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007a. [p. 31, 32, 33, 37, 58, 73, and 92]

F. Fouss, L. Yen, A. Pirotte, and M. Saerens. An experimental investigation of seven kernels on two collaborative recommendation tasks. *Submitted for publication*, 2007b. [p. 13, 15, 29, 30, 31, 33, 55, 67, 69, and 73]

G

Eugene Garfield. Agony and the ecstasyÑthe history and meaning of the journal impact factor. In *Proceedings of the International Congress on Peer Review and Bibliomedical Publication*, Chicago, 2005. [p. 25]

Christian Genest and James V. Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 36:114–148, 1986. [p. 109]

L. Getoor and B. Taskar. *Introduction to statistical relational learning*. The MIT Press, 2007. [p. 66]

F. Gobel and A. A. Jagers. Random walks on graphs. *Stochastic Processes and their Applications*, 2:311–336, 1974. [p. 36]

Gene H. Golub and Charles F. Van Loan. *Matrix Computations, 3th Ed.* Johns Hopkins University Press, 1996. [p. 45 and 72]

M. Gori and A. Pucci. Research paper recommender systems: A random-walk based approach. *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006a. [p. 34 and 35]

M. Gori and A. Pucci. A random-walk based scoring algorithm with application to recommender systems for large-scale e-commerce. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006b. [p. 34 and 35]

C. Greenhalgh and M. Rogers. The value of innovation: The interaction of competition, R&D and IP. *Research Policy*, 35(4):562–580, 2006. [p. 132]

Z. Griliches. Patent statistics as economic indicators: A survey. *Journal of Economic Literature*, 28, 1990. [p. 132]

H

B. Hall, A. Jaffe, and M. Trajtenberg. Market value and patent citations. *RAND Journal of Economics*, 36:16–38, 2005. [p. 83 and 86]

B.H. Hall. The patent paradox revisited: an empirical study of patenting in the US semiconductor industry, 1979-1995. *RAND Journal of Economics*, 32(1):101–128, 2001. [p. 83, 86, and 132]

Jihun Ham, Daniel Lee, Sebastian Mika, and Bernhard Scholkopf. A kernel view of the dimensionality reduction of manifolds. *Proceedings of the 21st International Conference on Machine Learning (ICML2004)*, 2004. [p. 33 and 55]

David Harel and Yehuda Koren. On clustering using random walks. *Proceedings of the conference on the Foundations of Software Technology and Theoretical Computer Science; Lecture Notes in Computer Science*, 2245:18–41, 2001. [p. 36]

D. Harhoff, G. von Graevenitz, and S. Wagner. European Patent Thickets: Do They Exist, Do They Matter. In *Second EPIP Conference, Lund*, 2007. [p. 132]

David A. Harville. *Matrix Algebra from a Statistician's Perspective*. Springer-Verlag, 1997. [p. 133]

Mark Herbster, Massimiliano Pontil, and Sergio Rojas-Galeano. Fast prediction on a tree. *Proceedings of the 22th Neural Information Processing conference (NIPS 2008)*, pages 657–664, 2008. [p. 91]

H. Hirai, K. Murota, , and M. Rikitoku. Svm kernel by electric network. *Pacific Journal of Optimization*, 1:509–526, 2005. [p. 32 and 92]

X. Hu, X. Zhang, C. Lu, EK Park, and X. Zhou. Exploiting Wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 389–396. ACM, 2009. [p. 109, 128, and 131]

I

T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto. Application of kernels to link analysis. *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 586–592, 2005. [p. 30, 67, 69, and 73]

J

Robert A. Jacobs. Methods for combining experts' probability assessments. *Neural Computation*, 7:867–888, 1995. [p. 109]

A. Jaffe, M. Trajtenberg, and R. Henderson. Geographic localization of knowledge spillovers as evidenced by patent citations. *The Quarterly Journal of Economics*, 108(3), 1993. [p. 85 and 132]

E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106: 620–630, 1957. [p. 21 and 38]

Frederick Jelinek. *Statistical methods for speech recognition*. The MIT Press, 1997. [p. 17 and 79]

T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 143–151. Citeseer, 1997. [p. 96]

T. Joachims. Transductive learning via spectral graph partitioning. In *Proc. of the* 20^{th} *International Conference on Machine Learning*, Washington DC, 2003. [p. 90]

T. Joachims, T.J. De, N. Cristianini, N. Uk, and R. Ac. Composite kernels for hypertext categorisation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001. [p. 127]

I. Jolliffe. *Principal components analysis, 2th ed.* Springer-Verlag, 2002. [p. 14]

D. Jungnickel. *Graphs, networks, and algorithms, 2th ed.* Algorithms and Computation in Mathematics, 2005. [p. 7]

Daniel Jurafsky and James Martin. *Speech and language processing*. Prentice-Hall, 2000. [p. 17]

K

J. Kandola, N. Cristianini, and J. Shawe-Taylor. Learning semantic similarity. *Advances in Neural Information Processing Systems*, pages 657–664, 2002. [p. 36]

A. Kapoor, Y. Qi, H. Ahn, and R. Picard. Hyperparameter and kernel learning for graph based semi-supervised classification. In *Advances in NIPS.*, 2005. [p. 90]

J. N. Kapur and H. K. Kesavan. *Entropy optimization principles with applications.* Academic Press, 1992. [p. 21]

John G. Kemeny and J. Laurie Snell. *Finite Markov Chains.* Springer-Verlag, 1976. [p. 8, 13, and 23]

M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25, 1963. [p. 28]

J. Kittler and F. M. Alkoot. Sum versus vote fusion in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):110–115, 2003. [p. 109]

Josef Kittler, Mohamad Hatef, Robert Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998. [p. 109]

D. J. Klein and M. Randic. Resistance distance. *Journal of Mathematical Chemistry*, 12:81–95, 1993. [p. 31, 32, and 36]

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. [p. 25]

George J. Klir and Tina A. Folger. *Fuzzy sets, uncertainty, and information.* Prentice-Hall, 1988. [p. 109]

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002. [p. 36 and 90]

Y. Koren, S. North, and C. Volinsky. Measuring and extracting proximity in networks. *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 245–255, 2006. [p. 33]

Y. Koren, S. North, and C. Volinsky. Measuring and extracting proximity graphs in networks. *ACM Transactions on Knowledge Discovery in Data*, 1(3):12:1–12:30, 2007. [p. 33]

L

Frank Lad. *Operational subjective statistical methods*. John Wiley & Sons, 1996. [p. 109]

S. Lafon and A. B. Lee. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, 2006. [p. 37]

J. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007. [p. 14]

P. Lehwark, U. Brefeld, and K.R. M
"uller. Contextualized text classification using wikipedia. 2010. [p. 109, 128, and 131]

William B. Levy and Hakan Delic. Maximum entropy aggregation of individual opinions. *IEEE Transactions on Systems, Man and Cybernetics*, 24(4):606–613, 1994. [p. 109]

David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58 (7):1019–1031, 2007. [p. 36, 74, and 90]

F. Lichtenberg and B. van Pottelsberg de la Potterie. International r&d spillovers: A comment. *European Economic Review*, 42(8), 1996. [p. 85]

L. Lovasz. Random walks on graphs: A survey. *Combinatorics: Paul Erdos is eighty*, 2:353–397, 1996. [p. 36]

W. Lu, J. Janssen, E. Milos, N. Japkowicz, and Y. Zhang. Node similarity in the citation graph. *Knowledge and Information Systems*, 11(1):105–129, 2006. [p. 37]

M

S. Macskassy and F. Provost. A simple relational classifier. *Proceedings of the Workshop on Multi-Relational Data Mining in conjunction with KDD-2003 (MRDM-2003)*, 2003. [p. 66]

S.A. Macskassy. Improving learning in networked data by combining explicit and mined links. In *Proceedings of the National Conference On Artificial Intelligence*, volume 22, page 590. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007. [p. 95, 101, 102, 103, 104, 105, 106, and 128]

Sofus A. Macskassy and Foster Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983, 2007. [p. 56, 58, 59, 65, 66, 67, 81, 84, and 128]

Francis Maes, Stéphane Peters, Ludovic Denoyer, and Patrick Gallinari. Simulated iterative classification: A new learning procedure for graph labeling. In *European Conference on Machine Learning*, Bled, Slovenia, 2009. [p. 66]

Christopher Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to information retrieval*. Cambridge University Press, 2008. [p. 14]

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. Academic Press, 1979. [p. 13, 14, and 15]

Christopher Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36:226–239, 1999. [p. 109]

Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. The Society for Industrial and Applied Mathematics (SIAM), 2001. [p. 45]

Tom M. Mitchell. *Machine learning*. McGraw-Hill Compagnies, 1997. [p. 96]

B. Mohar. Laplace eigenvalues of graphs – a survey. *Discrete Mathematics*, 109: 171–183, 1992. [p. 33]

K. Murphy, Y. Weiss, and M.I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of Uncertainty in AI*, pages 467–475. Citeseer, 1999. [p. 66]

In Jae Myung, Sridhar Ramamoorti, and Jr Andrew D. Bailey. Maximum entropy aggregation of expert predictions. *Management Science*, 42(10):1420–1436, 1996. [p. 109]

N

Boaz Nadler, Stephane Lafon, Ronald Coifman, and Ioannis Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Advances in Neural Information Processing Systems (NIPS) 18*, pages 955–962, 2005. [p. 37 and 58]

Boaz Nadler, Stephane Lafon, Ronald Coifman, and Ioannis Kevrekidis. Diffusion maps, spectral clustering and reaction coordinate of dynamical systems. *Applied and Computational Harmonic Analysis*, 21:113–127, 2006. [p. 37 and 58]

J. Neville and D. Jensen. Iterative classification in relational data. In *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000. [p. 66]

M.E.J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27 (1):39–54, 2005. [p. 26 and 27]

B. Noble and J. Daniels. *Applied linear algebra, 3th ed.* Prentice-Hall, 1988. [p. 13]

J. R. Norris. *Markov Chains*. Cambridge University Press, 1997. [p. 8 and 13]

O

European Patent Office. *PatStat*. Statistical Patent Database (PatStat), 2007. [p. 84]

H.J. Oh, S.H. Myaeng, and M.H. Lee. A practical hypertext catergorization method using links and incrementally available class information. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 264–271. ACM, 2000. [p. 128]

P

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. *Technical Report 1999-0120, Computer Science Department, Stanford University*, 1999. [p. 4, 33, 36, 55, and 74]

C.R. Palmer and C. Faloutsos. Electricity based external similarity of categorical attributes. *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'03)*, pages 486–500, 2003. [p. 36]

J.-Y. Pan, H.-J. Yang, C. Faloutsos, and P. Duygulu. Automatic multimedia cross-modal correlation discovery. *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 82(4):331–338, 2006. [p. 33, 35, 55, 67, 73, and 74]

P. Pons and M. Latapy. Computing communities in large networks using random walks. *Proceedings of the International Symposium on Computer and Information Sciences (ISCIS2005). Lecture Notes in Computer Science, Springer-Verlag*, 3733:284–293, 2005. [p. 37 and 58]

P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006. [p. 37 and 58]

A. Pothen, H. D. Simon, and K-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990. [p. 33]

Q

Huaijun Qiu and Edwin R. Hancock. Image segmentation using commute times. *Proceedings of the 16th British Machine Vision Conference (BMVC 2005)*, pages 929–938, 2005. [p. 33]

Huaijun Qiu and Edwin R. Hancock. Clustering and embedding using commute times. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007. [p. 33]

R

L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–285, 1989. [p. 17, 75, 77, and 79]

D. Rao, D. Yarowsky, and C. Callison-Burch. Affinity measures based on the graph Laplacian. In *Proceedings of the 3rd Textgraphs Workshop on Graph-Based Algorithms for Natural Language Processing*, pages 41–48. Association for Computational Linguistics, 2008. [p. 69 and 73]

L.E. Reichl. *A modern course in statistical physics, 2nd ed.* Wiley, 1998. [p. 21 and 38]

Volker Roth, Julian Laub, Joachim Buhmann, and Klaus-Robert Muller. Going metric: Denoising pairwise data. 2002. [p. 13]

P. Rousseuw and G. Molenberghs. Transformation of non positive semidefinite correlation matrices. *Communications in Statistics, Theory and Methods*, 22(4):965Ð984, 1993. [p. 13]

S

M. Saerens and F. Fouss. Yet another method for combining classifiers outputs: A maximum entropy approach. *Multiple Classifier Systems*, pages 82–91, 2004. [p. 109]

Marco Saerens, Francois Fouss, Luh Yen, and Pierre Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning (ECML 2004). Lecture Notes in Artificial Intelligence, vol. 3201, Springer-Verlag, Berlin*, pages 371–383, 2004. [p. 31, 32, 36, 58, 73, and 92]

Marco Saerens, Youssef Achbany, Francois Fouss, and Luh Yen. Randomized shortest-path problems: Two related models. *Neural Computation*, 21(8):2363–2404, 2009. [p. 4, 18, 20, 21, 68, 75, 76, 77, 91, and 130]

P. Sarkar and A. Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007. [p. 33, 73, 81, 90, and 91]

P. Sarkar, A. Moore, and A. Prakash. Fast incremental proximity search in large graphs. *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 896–903, 2008. [p. 90]

B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 5(10):1299–1319, 1998. [p. 13 and 15]

Bernhard Scholkopf and Alexander Smola. *Learning with kernels*. The MIT Press, 2002. [p. 13]

E. Schrodinger. *Statistical thermodynamics, 2nd ed.* Cambridge University Press, 1952. [p. 21 and 38]

F. Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002. [p. 127]

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004. [p. 13, 14, 15, and 36]

V. Sindhwani, P. Niyogi, and M. Belkin. Beyond the point cloud: From transductive to semi-supervised learning. *Proceedings of the International Conference on Machine Learning*, 2005. [p. 55, 61, and 66]

S. Slattery and T. Mitchell. Discovering test set regularities in relational domains. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 895–902. Citeseer, 2000. [p. 127]

H. Small. Co-citation in the scientific literature: a new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973. [p. 28]

T. F. Smith and M. S. Watterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981. [p. 17]

Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs. In Manfred Warmuth and Bernhard Schölkopf, editors, *Proceedings of the Conference on Learning Theory (COLT)*, pages 144–158, 2003. [p. 30, 36, and 90]

Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processiong Systems*, volume 14, Vancouver, Canada, 2001. MIT Press. [p. 69 and 90]

T

A. Tahbaz and A. Jadbabaie. A one-parameter family of distributed consensus algorithms with boundary: from shortest paths to mean hitting times. In *Proceedings of IEEE Conference on Decision and Control*, pages 4664–4669, 2006. [p. 37]

H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. *Proceedings of sixth IEEE International Conference on Data Mining (ICDM 2006)*, pages 613–622, 2006. [p. 33, 35, and 91]

H. Tong, Y. Koren, and C. Faloutsos. Fast direction-aware proximity for graph mining. *Proceedings of the ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD)*, pages 747–756, 2007. [p. 33, 67, 73, and 91]

H. Tong, C. Faloutsos, and J.-Y. Pan. Random walk with restart: fast solutions and applications. *Knowledge and Information Systems*, 14(3):327–346, 2008. [p. 33, 34, 35, 67, and 91]

M. Trajtenberg. A penny for your quotes: Patent citations and the value of innovations. *The RAND Journal of Economics*, 21:171–187, 1990. [p. 86]

V

N. van Zeebroeck, B. van Pottelsberghe de la Potterie, and W. Han. Issues in measuring the degree of technological specialization with patent data. *Scientometrics*, 66:481–492, 2006. [p. 85]

V. Vapnik. *Statistical Learning Theory*. Wiley, 1998. [p. 49 and 60]

W

Jung Wang, Tony Jebara, and Shih-Fu Chang. Graph tranduction via alternating minimization. *Proceedings of the Twenty-first International Conference on Machine Learning*, 2008. [p. 55, 61, 69, and 90]

S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994. [p. 18, 25, 27, 28, 75, and 92]

Michael C. Wendl. H-index: however ranked, citations need context. *Annals of the Institute of Statistical Mathematics*, 449(7161), 2007. [p. 25]

D.B. West et al. *Introduction to graph theory*. Prentice Hall Upper Saddle River, NJ, 2001. [p. 7]

Scott White and Padhraic Smyth. Algorithms for estimating relative importance in networks. *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 266–275, 2003. [p. 36]

G. Wu, E. Chang, and Z. Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, 2005. [p. 13]

Y

Yasutoshi Yajima and Tien-Fang Kuo. Efficient formulations for 1-svm and their application to recommendation tasks. *Journal of Computers*, 1(3):27–34, 2006. [p. 69 and 90]

L. Yen, F. Fouss, C. Decaestecker, P. Francq, and Marco Saerens. Graph nodes clustering based on the commute-time kernel. In *Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2007). Lecture notes in Computer Science*, volume LNAI4426, pages 1037–1045, 2007. [p. 15 and 33]

L. Yen, A. Mantrach, M. Shimbo, and M. Saerens. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *Proceedings of the 14th SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–793, 2008. [p. 33, 58, 77, and 91]

L. Yen, F. Fouss, C. Decaestecker, P. Francq, and Marco Saerens. Graph nodes clustering with the sigmoid commute-time kernel: a comparative study. *Data & Knowledge Engineering*, 2009. [p. 33]

Luh Yen, Denis Vanvyve, Fabien Wouters, Francois Fouss, Michel Verleysen, and Marco Saerens. Clustering using a random walk-based distance measure. In *Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN2005)*, pages 317–324, 2005. [p. 15 and 33]

S. Yu. *Kernel-based data fusion for machine learning: methods and applications in bioinformatics and text mining*. PhD thesis, K.U.L. [p. 109]

Z

Deli Zhao, Z.C. Lin, and X.O. Tang. Contextual distance for data perception. *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007. [p. 37]

D. Zhou and B. Scholkopf. Learning from labeled and unlabeled data using random walks. *Proceedings of the 26th DAGM Symposium, (Eds.) Rasmussen*, pages 237–244, 2004. [p. 4, 70, and 90]

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. *Proceedings of the Neural Information Processig Systems Conference (NIPS2003)*, pages 237–244, 2003. [p. 2, 51, 52, 60, 61, 66, 67, 68, 69, 90, and 130]

D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning*, pages 1041–1048, 2005. [p. 2, 4, 37, 56, 58, 59, 67, 68, 69, 70, 81, 90, and 130]

H. Zhou. Network landscape from a brownian particle perspective. *Physical Review E*, 67(041908), 2003a. [p. 33]

H. Zhou. Distance, dissimilarity index, and network community structure. *Physical Review E*, 67(061901), 2003b. [p. 33]

S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, page 494. ACM, 2007. [p. 127]

X. Zhu. Semi-supervised learning literature survey. In *http://pages.cs.wisc.edu/ jerryzhu/research/ssl/semireview.html*, 2008. [p. 49, 66, 67, and 90]

X. Zhu and A. Goldberg. *Introduction to semi-supervised learning*. Morgan & Claypool Publishers, 2009. [p. 49, 66, and 90]

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pages pp.912–919, 2003. [p. 51, 60, 61, 66, and 69]

X. Zhu, J. Kandola, J. Lafferty, and Z. Ghahramani. Graph kernels by spectral transforms. In *Semi-supervised learning, O. Chapelle, B. Scholkopf and A. Zien (editors)*, pages 277–291. MIT Press, 2006. [p. 36]