Semi-supervised Classification from Discriminative Random Walks

Conference Paper · August 2008

DOI: 10.1007/978-3-540-87479-9_29 · Source: DBLP

Marco Saerens
Université Catholique de Louvain - UCLouvain
128 PUBLICATIONS **3,620** CITATIONS

Pierre Dupont
Université Catholique de Louvain - UCLouvain
108 PUBLICATIONS **2,588** CITATIONS

Semi-supervised Classification from Discriminative Random Walks*

Jérôme Callut[1,2] , Kevin Françoisse[1,2], Marco Saerens[1,2], and Pierre Dupont[1,3]

[1] UCL Machine Learning Group (MLG)
[2] Louvain School of Management, ISYS unit, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium
{Jerome.Callut, Kevin.Francoisse, Marco.Saerens}@uclouvain.be
[3] Department of Computing Science and Engineering, INGI, Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium
Pierre.Dupont@uclouvain.be

Abstract This paper describes a novel technique, called $\mathcal{D}$-walks, to tackle semi-supervised classification problems in large graphs. We introduce here a betweenness measure based on passage times during random walks of bounded lengths. Such walks are further constrained to start and end in nodes within the same class, defining a distinct betweenness for each class. Unlabeled nodes are classified according to the class showing the highest betweenness. Forward and backward recurrences are derived to efficiently compute the passage times. $\mathcal{D}$-walks can deal with directed or undirected graphs with a linear time complexity with respect to the number of edges, the maximum walk length considered and the number of classes. Experiments on various real-life databases show that $\mathcal{D}$-walks outperforms NetKit [5], the approach of Zhou and Schölkopf [15] and the regularized laplacian kernel [2]. The benefit of $\mathcal{D}$-walks is particularly noticeable when few labeled nodes are available. The computation time of $\mathcal{D}$-walks is also substantially lower in all cases.

1 Introduction

Mining and learning problems involving structured data such as graphs, trees or sequences have received much attention recently. The present work is concerned with semi-supervised classification of nodes in a graph. Given an input graph with some nodes being labeled, the problem is to predict the missing node labels. This problem has numerous applications such as classification of individuals in social networks, linked documents (*e.g.* patents or scientific papers) categorization or protein function prediction, to name a few. Even when the data is not

* Part of this work was supported by the STRATEGO project funded by the Region wallonne, Belgium.

initially structured as a graph, it can be convenient to build a neighborhood graph of similar examples from an affinity matrix.

Several approaches have been proposed to tackle semi-supervised classification problems in graphs. Kernel methods [12,15,14] embed the nodes of the input graph into an Euclidean feature space where a decision boundary can be estimated. For instance, the methods proposed in [15] rely on a kernel obtained by computing the inverse of a regularized graph Laplacian matrix. Such a kernel can be interpreted in terms of *commute times* during random walks performed on the graph. The method proposed in the present paper is related to the former approach, however it relies on *passage times* during random walks rather than on commute times. Despite their good predictive performance, kernel methods on graphs cannot easily scale up to large problems due to their high time complexity. NetKit [5] is an alternative relational learning approach. This general framework builds a model based on three components: a local classifier to generate class-priors, a relational classifier, which relies on the relations in the network to guess the class membership, and a so-called collective inferencing component which further refines the class predictions. The main advantage of this framework is that each of the three components can be instantiated with various existing methods making it easily adaptable to many situations. This flexibility comes however with a time-consuming tuning process to optimize performance. Compared to the above mentioned kernel methods, it provides good performance while having a lower time complexity.

The approach proposed in this paper, called $\mathcal{D}$-walks (discriminative random walks), relies on random walks performed on the input graph seen as a Markov chain. More precisely, a betweenness measure, based on passage times during random walks of bounded length, is derived for each class (or label category). Unlabeled nodes are assigned to the category for which the betweenness is the highest. The $\mathcal{D}$-walks approach has the following properties: (i) it has a linear time complexity with respect to the number of edges, the maximum walk length considered and the number of classes; such a low complexity allows to deal with very large graphs, (ii) it can handle directed or undirected graphs, (iii) it can deal with multi-class problems and (iv) it has a unique hyper-parameter, the walk length, that can be tuned efficiently. Moreover, an extension of the technique is proposed to handle descriptive features on nodes (*e.g.* in social networks, such features could be the age or the number of children of individuals) via a similarity function.

This paper is organized as follows. Section 2 reviews basic notions about discrete time Markov chain and passage times during random walks. Section 3 introduces the $\mathcal{D}$-walks approach for semi-supervised classification in graphs. Section 4 introduces an extension to incorporate node features into the $\mathcal{D}$-walks classification technique. Links and differences between the $\mathcal{D}$-walks method and alternative approaches based on random walks are further described in section 5. Finally, section 6 reports comparative experimental results on real-life data.

2 Background

The semi-supervised node classification technique proposed in section 3 is based on *passage times* during random walks on the input graph seen as a *Markov chain* (MC). These classical notions are briefly reviewed in the present section. For a more detailed introduction to the MC theory, the reader is referred to the classical text books [4,7].

Definition 1 (Discrete time Markov Chain (MC)). *A* discrete time Markov Chain (MC) *is a stochastic process* $\{X_t \mid t \in \mathbb{N}\}$ *where the random variable* X *takes its value at any discrete time* t *in a countable set* $\mathcal{N}$ *and such that:*

$$P[X_t = q \mid X_{t-1}, X_{t-2}, \ldots, X_0] = P[X_t = q \mid X_{t-1}]. \tag{1}$$

This condition states that the probability of the next outcome only depends on the last value of the process (order 1 Markov property).

A finite MC can be represented by a 3-tuple $T = \langle \mathcal{N}, P, \boldsymbol{\iota} \rangle$ where $\mathcal{N}$ is a finite set of states with $n = |\mathcal{N}|$, P is a $n \times n$ row-stochastic matrix encoding the (homogeneous) transition probabilities $p_{qq'} = P[X_t = q' \mid X_{t-1} = q]$ for all $q, q' \in \mathcal{N}$ and $\boldsymbol{\iota}$ is an n-dimensional vector representing the initial probability distribution, *i.e.* $\iota_q = P[X_0 = q]$ for all $q \in \mathcal{N}$.

A *random walk* on a MC can be defined as follows: a random walker starts in a state q according to the initial distribution $\boldsymbol{\iota}$. Next, he moves to some state $q' \in \mathcal{N}$ according to the transition probability matrix P. Repeating this last operation $l \in \mathbb{N}$ times results in a l-step random walk. In a MC, a state q is said to be *absorbing* if there is a probability 1 to go from q to itself. In other words, once an absorbing state has been reached in a random walk, the walker will stay on this state forever. A MC for which there is a probability 1 to end up in an absorbing state is called an *absorbing MC*. In such a model, the state set can be divided into the absorbing state set $\mathcal{N}_A$ and its complementary set, the transient state set $\mathcal{N}_T = \mathcal{N} \setminus \mathcal{N}_A$. The *passage time* function counts the number of times a given node has been visited during a random walk.

Definition 2 (Passage Time). *Given a MC,* $T = \langle \mathcal{N}, P, \boldsymbol{\iota} \rangle$, *the* passage time *is a function* $\mathrm{pt} : \mathcal{N} \times \mathcal{N} \rightarrow \mathbb{N}$ *such that* $\mathrm{pt}(q)$ *is the number of times the process reaches the state* q*:*

$$\mathrm{pt}(q) = |\{t \in \mathbb{N} \mid X_t = q\}| \tag{2}$$

The *mean passage time* (MPT) denotes the expectation of this quantity: $\mathbb{E}[\mathrm{pt}(q)]$. The MPT is clearly infinite for absorbing states. For transient states, the MPT can be obtained by computing the so-called *fundamental matrix*: $N = (I - P_T)^{-1}$ where I is the $|\mathcal{N}_T| \times |\mathcal{N}_T|$ identity matrix and P_T is the transition probability matrix restricted to transient states (subscript T). The entry $n_{q'q}$ contains the MPT in state $q \in \mathcal{N}_T$ during random walks starting in state q'. Hence, $\mathbb{E}[\mathrm{pt}(q)] = [\boldsymbol{\iota}_T' N]_q$ where $\boldsymbol{\iota}_T'$ is the (transpose of the) initial probability vector reduced to transient states[1]. It should be stressed that this expectation is taken over random walks of any (positive) length. In contrast, the betweenness proposed in section 3.3 relies on passage times computed for walks of bounded length.

3 Discriminative Random Walks

This section presents the $\mathcal{D}$-walks approach to perform semi-supervised classification of nodes in a graph. This technique is based on a betweenness measure computed from passage times during random walks performed in the input graph. The problem statement is detailed in section 3.1. A betweenness measure using random walks of unbounded length is introduced in section 3.2. Section 3.3 refines this measure by considering bounded walks up to a prescribed length. Section 3.4 shows how to classify unlabeled nodes based on the proposed betweenness measure.

3.1 Problem Statement

Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ denote an input graph where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. In the sequel, $n = |\mathcal{N}|$ denotes the number of nodes and $m = |\mathcal{E}|$ the number of edges in the graph. Let also A denote the $n \times n$ adjacency matrix of $\mathcal{G}$. Since the graph $\mathcal{G}$ may be directed and weighted, the matrix A is not necessarily symmetric nor binary. Furthermore, it is assumed that A is an affinity matrix: the edge weights are positive and the higher the value, the easier the connection between the corresponding nodes. The graph $\mathcal{G}$ is assumed partially labeled. The nodes in the *labeled set* $\mathcal{L} \subset \mathcal{N}$ are assigned to a category from a discrete set $\mathcal{Y}$. The *unlabeled set* is defined as $\mathcal{U} = \mathcal{N} \setminus \mathcal{L}$. The label of a node $q \in \mathcal{L}$ is written y_q and $\mathcal{L}_y$ denotes the set of nodes in class y, with $n_y = |\mathcal{L}_y|$. A local consistency of the node labeling is assumed, *i.e.* nodes within a neighborhood are likely to share the same label. The task is to classify unlabeled nodes in the graph.

3.2 $\mathcal{D}$-Walks Betweenness

Random walks on a graph can be modeled by a discrete-time Markov chain $\{X_t \in \mathcal{N}\}_{t \in \mathbb{N}}$ (MC) describing the sequence of nodes visited during the walk. The random variable X_t represents the state of the MC reached at the discrete time index t. Since each state of the Markov chain corresponds to a distinct node of the graph, the state set of the MC is simply the set of nodes $\mathcal{N}$. The terms *nodes* and *states* are thus used interchangeably in this paper. The transition probability to state q' at time t, given that the last state is q at time $t-1$, is defined as:

$$P[X_t = q' \mid X_{t-1} = q] = p_{qq'} \triangleq \frac{a_{qq'}}{\sum_{q' \in \mathcal{N}} a_{qq'}}. \tag{3}$$

[1] It is assumed here that random walks cannot start in an absorbing state.

Thus, from any state q, the probability to jump to state q' is proportional to the weight $a_{qq'}$ of the edge from q to q' (and then normalized). These transition probabilities are stored in an $n \times n$ transition matrix $P = \{p_{qq'}\}_{q,q' \in \mathcal{N}}$.

We introduce *discriminative random walks* ($\mathcal{D}$-walks, for short) to define a betweenness measure used for node classification (see section 3.4). A $\mathcal{D}$-walk is a random walk starting in a labeled node and ending when any node having the same label (possibly the starting node itself) is reached for the first time.

Definition 3 ($\mathcal{D}$-walks). *Given a MC defined on the state set $\mathcal{N}$ and a class $y \in \mathcal{Y}$, a $\mathcal{D}$-walk is a sequence of state $q_0, \ldots, q_l$ such that $y_{q_0} = y_{q_l} = y$ and $y_{q_t} \neq y$ for all $0 < t < l$.*

The notation $\mathcal{D}^y$ refers to the set of all $\mathcal{D}$-walks starting and ending in a node of class y.

The betweenness function $B(q, y)$ measures how much a node $q \in \mathcal{U}$ is located "between" nodes of class $y \in \mathcal{Y}$. The betweenness $B(q, y)$ is formally defined as the expected number of times node q is reached during $\mathcal{D}^y$-walks.

Definition 4 ($\mathcal{D}$-walks betweenness). *Given an unlabeled node $q \in \mathcal{U}$ and a class $y \in \mathcal{Y}$, the $\mathcal{D}$*-walks betweenness function $\mathcal{U} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ *is defined as follows:*

$$B(q, y) \triangleq \mathbb{E}\left[\text{pt}(q) \mid \mathcal{D}^y\right] \tag{4}$$

This betweenness measure is closely related to the one proposed by Newman in [6]. Our measure is however relative to a specific class y rather than to the whole graph, which is more informative in the context of classification. $B(q, y)$ can be computed using standard absorbing MC techniques (see section 2). Nodes belonging to $\mathcal{L}_y$ are first duplicated such that the original nodes are used as absorbing states and the duplicated ones as starting states. The transition matrix P is augmented as follows: (i) one duplicates the rows of P corresponding to nodes in $\mathcal{L}_y$ at the bottom of the matrix, (ii) one adds n_y columns full of zeroes[2] at the right of P and (iii) one defines $p_{qq'} = 1 \iff q' = q$ and 0 otherwise, for all $q \in \mathcal{L}_y$. The augmented matrix is denoted here by ${}^yP$. The initial distribution vector is adapted accordingly, resulting in the vector ${}^y\boldsymbol{\iota}$. The betweenness is finally computed as follows:

$$B(q, y) = [{}^y\boldsymbol{\iota}_T'(I - {}^yP_T)^{-1}]_q \tag{5}$$

where ${}^yP_T$ and ${}^y\boldsymbol{\iota}_T$ respectively denote the transition matrix and the initial distribution vector restricted to transient states (*i.e.* all states but those in $\mathcal{L}_y$). The betweenness computation has a computational complexity $\mathcal{O}(n^3)$ due to the matrix inversion. The space complexity is $\mathcal{O}(n^2)$ since the inverse matrix is generally dense even for sparse input graphs. Such high complexities makes the technique only tractable for graphs containing a few thousands nodes. Optimized matrix inversion, possibly relying on a spectral decomposition, can be used to

[2] The n_y added states are only used to start walks and cannot be reached from any node in the graph.

tackle this problem. We propose an alternative approach, detailed in the next section, by adapting the betweenness definition. It offers a much lower time and space complexity algorithm, which also proved to be more accurate for semi-supervised classification.

3.3 Bounded $\mathcal{D}$-Walks Betweenness

This section describes a betweenness measure based on $\mathcal{D}$-walks of bounded length. The notation $\mathcal{D}_l^y$ refers to the set of all $\mathcal{D}$-walks of length *exactly equal* to l, starting and ending in a node of class y. We also consider $\mathcal{D}_{\leq L}^y$ referring to the set of all bounded $\mathcal{D}$-walks *up to* a given length L. The bounded $\mathcal{D}$-walks betweenness measure $B_L(q, y)$ is formally defined hereafter.

Definition 5 (Bounded $\mathcal{D}$-walks betweenness). *Given an unlabeled node $q \in \mathcal{U}$ and a class $y \in \mathcal{Y}$, the* $\mathcal{D}$-walks betweenness function $\mathcal{U} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ *is defined as follows:*

$$B_L(q, y) \triangleq \mathbb{E}\left[\text{pt}(q) \mid \mathcal{D}_{\leq L}^y\right]$$

Bounding the walk length has two major benefits: (i) better classification results are systematically obtained with respect to unbounded walks[3] in our experiments, (ii) the betweenness measure can be computed very efficiently. The unbounded betweenness can also be approximated with the bounded one by considering large but finite L values. More precisely, it can be shown that the bounded betweenness converges *almost geometrically fast* with L towards the unbounded betweenness value [1].

An efficient betweenness computation can be achieved using forward and backward variables, similar to those used in the Baum-Welch algorithm for Hidden Markov Models (HMM) parameter estimation [9]. Given a state $q \in \mathcal{N}$ and a time $t \in \mathbb{N}^0$, the forward variable $\alpha^y(q, t)$ computes the probability to reach state q after t steps without passing through[4] nodes in class y, while starting initially from any state in class y. The forward variables are computed using the following recurrence:

$$\left| \begin{array}{ll} (\text{case } t = 1) & \alpha^y(q, 1) = \sum_{q' \in \mathcal{L}_y} \frac{1}{n_y} p_{q'q} \\ \\ (\text{case } t \geq 2) & \alpha^y(q, t) = \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \alpha^y(q', t-1)\, p_{q'q} \end{array} \right. \qquad (6)$$

It is assumed that walks can start in any state of class y with a uniform probability $1/n_y$. Notice that the case $t = 1$ allows transitions outgoing from a state in class y (*i.e.* a starting state) whereas such transitions are forbidden in the induction case ($t \geq 2$), as walks are stopped as soon as a node in class y is reached.

[3] The maximum walk length has to be chosen for instance by cross-validation (see section 6).

[4] In contrast with *leaving from* a node q, *passing through* q means to jump from some node q' to q and then to leave from q.

Given a state $q \in \mathcal{N}$ and a time $t \in \mathbb{N}^0$, the backward variable $\beta^y(q,t)$ computes the probability that state q is attained by the process t steps before reaching any node labeled y for the first time. The backward variables are computed using the following recurrence:

$$\left| \begin{array}{ll} (\text{case } t = 1) & \beta^y(q,1) = \sum_{q' \in \mathcal{L}_y} p_{qq'} \\ \\ (\text{case } t \geq 2) & \beta^y(q,t) = \sum_{q' \in \mathcal{N} \setminus \mathcal{L}_y} \beta^y(q', t-1)\, p_{qq'} \end{array} \right. \tag{7}$$

Transitions incoming to a state in class y are only allowed at the end of walks (case $t = 1$) since we are interested in walks ending when any node in $\mathcal{L}_y$ is reached for the first time. The time complexity of the forward and backward recurrences is $\Theta(mL)$, where m is the number of edges and L is the maximal walk length considered.

In order to compute $B_L(q,y)$, let us first calculate the MPT in a node $q \in \mathcal{U}$ during $\mathcal{D}_l^y$-walks (*i.e.* walks of length *exactly equal* to l): $\mathbb{E}\left[\text{pt}(q) \mid \mathcal{D}_l^y\right]$. The (length-conditionned) passage time function $\text{pt}(q)$ can be decomposed as a sum of indicator variables: $\text{pt}(q) = \sum_{t=1}^{l-1} \mathbb{I}\{X_t = q\}$. Consequently, the desired expectation is given by

$$\mathbb{E}\left[\text{pt}(q) \mid \mathcal{D}_l^y\right] = \mathbb{E}\left[\sum_{t=1}^{l-1} \mathbb{I}\{X_t = q\} \,\middle|\, \mathcal{D}_l^y\right] = \sum_{t=1}^{l-1} \mathbb{E}\left[\mathbb{I}\{X_t = q\} \mid \mathcal{D}_l^y\right] \tag{8}$$

$$= \sum_{t=1}^{l-1} P[X_t = q \mid \mathcal{D}_l^y] = \sum_{t=1}^{l-1} \frac{P[X_t = q \wedge \mathcal{D}_l^y]}{P[\mathcal{D}_l^y]} \tag{9}$$

The joint probability in the numerator of equation (9) can be computed as $P[X_t = q \wedge \mathcal{D}_l^y] = \alpha^y(q,t)\beta^y(q, l-t)$, that is the probability to start in any node of $\mathcal{L}_y$, to reach node q at time t and to complete the walk $l - t$ steps later. The probability to perform a $\mathcal{D}_l^y$-walk is obtained as $P[\mathcal{D}_l^y] = \sum_{q' \in \mathcal{L}_y} \alpha^y(q', l)$. Therefore, the desired expectation is given by

$$\mathbb{E}\left[\text{pt}(q) \mid \mathcal{D}_l^y\right] = \frac{\sum_{t=1}^{l-1} \alpha^y(q,t)\beta^y(q,l-t)}{\sum_{q' \in \mathcal{L}_y} \alpha^y(q',l)} \tag{10}$$

Finally, the betweenness measure based on walks *up to* length L is obtained as an expectation of the betweennesses for all length $1 \leq l \leq L$:

$$B_L(q,y) = \sum_{l=1}^{L} \frac{P[\mathcal{D}_l^y]}{Z}\, \mathbb{E}\left[\text{pt}(q) \mid \mathcal{D}_l^y\right] \tag{11}$$

$$= \frac{\sum_{l=1}^{L} \sum_{t=1}^{l-1} \alpha^y(q,t)\beta^y(q,l-t)}{\sum_{l=1}^{L} \sum_{q' \in \mathcal{L}_y} \alpha^y(q',l)} \tag{12}$$

$$= \frac{\left(\sum_{t=1}^{L} \alpha^y(q,t)\right)\left(\sum_{l=1}^{L-t} \beta^y(q,l)\right)}{\sum_{l=1}^{L} \sum_{q' \in \mathcal{L}_y} \alpha^y(q',l)} \tag{13}$$

The distribution $P[\mathcal{D}_l^y]$ is defined on all discrete times $t \in [1, \infty)$, however only walks up to length L are considered here. Therefore, we introduce a normalization constant Z ensuring that $\sum_{l=1}^{L} P[\mathcal{D}_l^y]/Z$ sums to one, making the expectation well-defined. Once the values of the forward and backward variables are stored in lattices, $B_L(q, y)$ can be obtained with a time complexity in $\Theta(L.n_y)$ by precomputing all the terms of the inner sum contained in the numerator of equation (13). The complexity for computing the betweenness for all unlabeled nodes with respect to a specific class y is thus $\Theta(m.L)$ as it is dominated by the cost of the recurrence computations. The space complexity is $\Theta(m + L.n)$, *i.e.* the space required to store the graph and the forward and backward lattices.

3.4 Classification Using the Betweenness

Unlabeled nodes are classified using a maximum *a posteriori* (MAP) decision rule from the betweenness computed for each class. The class-conditionned likelihood of a node q with respect to a specific class y is defined from its class betweenness as follows:

$$P[q \mid y] \triangleq \frac{B_L(q, y)}{\sum_{y' \in \mathcal{Y}} B_L(q, y')} \tag{14}$$

The label of a node $q \in \mathcal{U}$ is predicted by:

$$\hat{y}_q = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, P[q \mid y]\ P[y] \tag{15}$$

where $P[y]$ is estimated as the proportion of nodes belonging to class y. Taking the argmax on the class posteriors corresponds to a crisp node classification. A fuzzy node classification can be obtained by directly outputting the class posteriors. Finally, the time complexity to classify all unlabeled nodes using the bounded $\mathcal{D}$-walks betweenness is $\Theta(|\mathcal{Y}|.m.L)$ where m is the number of edges in the graph and L is the maximal walk length allowed. The space complexity is the same as for computing the betweenness relative to a specific class, *i.e.* $\Theta(m + L.n)$.

4 Incorporating Node Features

This section presents an extension of the $\mathcal{D}$-walks approach to take node features or attributes into account during classification. For instance, such features could be the age, the nationality or the marital status of individuals in a social network. Incorporating such features can clearly improve the classification performance as the system can discriminate using both structural and descriptive information on nodes. The feature vector corresponding to a node q is denoted by $\phi(q)$. We assume that a similarity function $k(\phi(q), \phi(q'))$ between feature vectors is available. The resulting $n \times n$ similarity matrix for all pairs of nodes is written K. Note that the similarity function needs not be a genuine Mercer kernel (*i.e.* matrix K is not required to be positive-definite). However, it is assumed that $k(\phi(q), \phi(q'))$ takes positive values.

Our approach for incorporating nodes features is based on a reweighting of the adjacency matrix. Each entry $a_{qq'}$ of the adjacency matrix is simply multiplied by the similarity between the concerned nodes $k_{q,q'}$. One of our modeling hypothesis (see section 3.1) states that the higher the edge weight the easier the connection. The edge reweighting proposed here is consistent with this assumption as the connection between two nodes will be reinforced proportionally to their feature-based similarity. In matrix form, the reweighted adjacency matrix ${}^{\phi}A$ is obtained as follows:

$$ {}^{\phi}A = A \circ K \tag{16} $$

where $\circ$ denotes the element-wise (Hadamard) product between matrices. This multiplicative reweighting preserves the sparseness of the graph and the similarity function needs only to be evaluated for pairs (q, q') such that $a_{qq'} > 0$. Consequently, the complexity of the $\mathcal{D}$-walks approach using node features is $\Theta(|\mathcal{Y}|.m.L + m.\text{Sim})$ where Sim is the time complexity of one similarity function evaluation. The space complexity remains unchanged with respect to the original $\mathcal{D}$-walks approach.

5 Some Links between $\mathcal{D}$-Walks and Related Approaches

It is worth stressing the links and differences between the $\mathcal{D}$-walks method described in section 3.3 and competing approaches. To simplify a bit the analysis we restrict first our attention to undirected graphs. For such graphs, the weighted adjacency matrix A is symmetric.

The method of Zhou and Schölkopf [15], compared experimentally with $\mathcal{D}$-walks in section 6, can also be interpreted in terms of random walks. This method computes the matrix $S = D^{-1/2}AD^{-1/2}$ where D denotes the diagonal matrix of node degrees: $d_q = \sum_{q \in \mathcal{N}} a_{qq'}$. A specific element of S is simply given by:

$$ s_{qq'} = \frac{a_{qq'}}{\sqrt{d_q}\sqrt{d_{q'}}} \tag{17} $$

In contrast with the transition probability matrix $P = D^{-1}A$, the S matrix is not necessarily row-stochastic. Obviously, $P = S$ whenever every node has the same degree. Semi-supervised classification in [15] relies on the inverse of a regularized and normalized[5] laplacian $\tilde{L}$:

$$ \tilde{L}^{-1} = (I - \alpha S)^{-1} \text{ with } 0 < \alpha < 1 \tag{18} $$

In particular, the classification rule for the unlabeled node q can be written:

$$ \hat{y}_q = \underset{y \in \mathcal{Y}}{\text{argmax}} \sum_{\{q' \mid y_{q'} = y\}} [\tilde{L}^{-1}]_{qq'}, \text{ where } y_{q'} \text{ is the label of node } q' \tag{19} $$

[5] The approach of Zhou et al [15] can be considered as a variant of [2,3] which uses the regularized laplacian matrix $D - \alpha A$ instead of the standard one. Performance of the later approach is also reported in our experiments.

Equation (18) can be reformulated using a Neumann series:

$$(I - \alpha S)^{-1} = \sum_{l=0}^{\infty} (\alpha S)^l = I + \alpha S + \alpha^2 S^2 + \dots \tag{20}$$

The first equality is valid only if the series converges[6]. Under this assumption, the sum of matrix powers in (20) is highly similar to those computed (however through recurrence relations) in the bounded $\mathcal{D}$-walks approach.

To push the analogy a bit further, let us consider a scaled transition probability matrix $\tilde{P} = \alpha P$. Such matrix defines a lazy random walk in the original graph since a transition from state q to a distinct state q' has a probability

$$\tilde{p}_{qq'} = \frac{\alpha \, a_{qq'}}{d_{q'}}, q \neq q' \tag{21}$$

and, consequently, the random walk remains on state q with probability $1 - \alpha$. Finally, equation (20) relies on the successive powers of αS and one specific element of this matrix is written:

$$\alpha s_{qq'} = \alpha \frac{a_{qq'}}{\sqrt{d_q}\sqrt{d_{q'}}} = \sqrt{\tilde{p}_{qq'} \tilde{p}_{q'q}} \tag{22}$$

One observes that the quantity of interest in this approach is the geometric mean of the transition probability in one direction and the opposite direction during such lazy random walks. This symmetric quantity is consistent with the commute times interpretation detailed in [15].

The unique hyper-parameter is here α which controls the degree of laziness of the random walks. In contrast, the unique hyper-parameter for the $\mathcal{D}$-walks method is the maximal walk length L, which corresponds to the maximal matrix power considered. This difference can be interpreted as a distinct regularization choice [10]. Note that bounding the maximal walk length may not be the most appropriate choice when defining a graph kernel. We argue that this choice is however better for the semi-supervised classification problem considered here, as confirmed experimentally in section 6.

Another important aspect of the $\mathcal{D}$-walks approach is to consider only walks that start and end in a labeled node of a given class but do not go through labeled nodes of the same class at intermediate steps. Intuitively, the random walks explain the connections with labeled nodes of a given class without diffusing unconditionally through the graph.

The method described in [15] was extended to directed graphs [14]. These approaches require that the random walk model has a unique stationary distribution. For directed graphs, this property can be satisfied by considering teleporting random walks as in [8]. The price to pay is the introduction of an additional hyper-parameter which defines the trade-off between the teleporting uniform probability and the natural transition probability derived from the adjacency matrix. The $\mathcal{D}$-walks approach does not require such an additional parameter

[6] This condition is satisfied if the spectral radius of αS is < 1.

to deal with directed graphs. This is an immediate consequence of bounding the walk length rather than relying on the asymptotic behavior of the walks.

An alternative semi-supervised classification method using random walks was proposed in [11]. This method considers walks starting from an unlabeled node and reaching a labeled node in exactly L steps. Again, nothing prevents the walks to go through labeled nodes of the same class during the intermediate steps. Szummer and Jaakkola also proposed the use of a different length parameter from each unlabeled node but this method did not pay off experimentally. The recurrences detailed in section 3.3 are also specific to the $\mathcal{D}$-walks approach. They are essential to guarantee a low computational complexity to be able to deal with large graphs.

6 Experiments

The experiments presented in this section have two goals. First, the classification performance of our approach is compared against competing methods. Secondly, the computing times on large-scale problems data are analyzed, demonstrating the efficiency and the scalability of the approach.

6.1 Data

The three case studies used for this paper are real-world representation of networked data. They all come from different domains that have been the subject of prior study in machine learning [5].

IMDb: The collaborative Internet Movie Database (`IMDb`) 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 graph contains 1196 movies which have the following class distribution (see Table 1):

Table 1. Class distribution for the `IMDb` data set

Category	**Size**
High-revenue	572
Low-revenue	597
Total	1169
Majority class accuracy	51.07%
Number of Edges	40564
Mean degree	36.02
Min degree	1
Max degree	181

Table 2. Class distribution for the `CORA` data set

Category	**Size**
Case-Based	402
Genetic Algorithms	551
Neural Networks	1064
Probabilistic Methods	529
Reinforcement Learning	335
Rule Learning	230
Theory	472
Total	3583
Majority class accuracy	29.7%
Number of Edges	22516
Mean degree	6.28
Min degree	1
Max degree	311

CORA: The `CORA` dataset contains computer science research papers. It includes the full citation graph as well as the topic of each paper as labels. Two papers are linked if one cites the other. The weight of an edge is generally one unless two papers cite each other, in which case it is two. The graph contains 3582 nodes with the following class distribution (see Table 2):

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, project, staff, and student. 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 3.

Table 3. Class distribution for the `WebKB` data set

	Size			
Category	**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 accuracy	41.9%	48.8%	39.2%	44.5%
Number of Edges	26832	32988	30462	33250
Mean degree	77.55	98.77	70.19	95.55
Min degree	1	1	1	1
Max degree	191	215	286	229

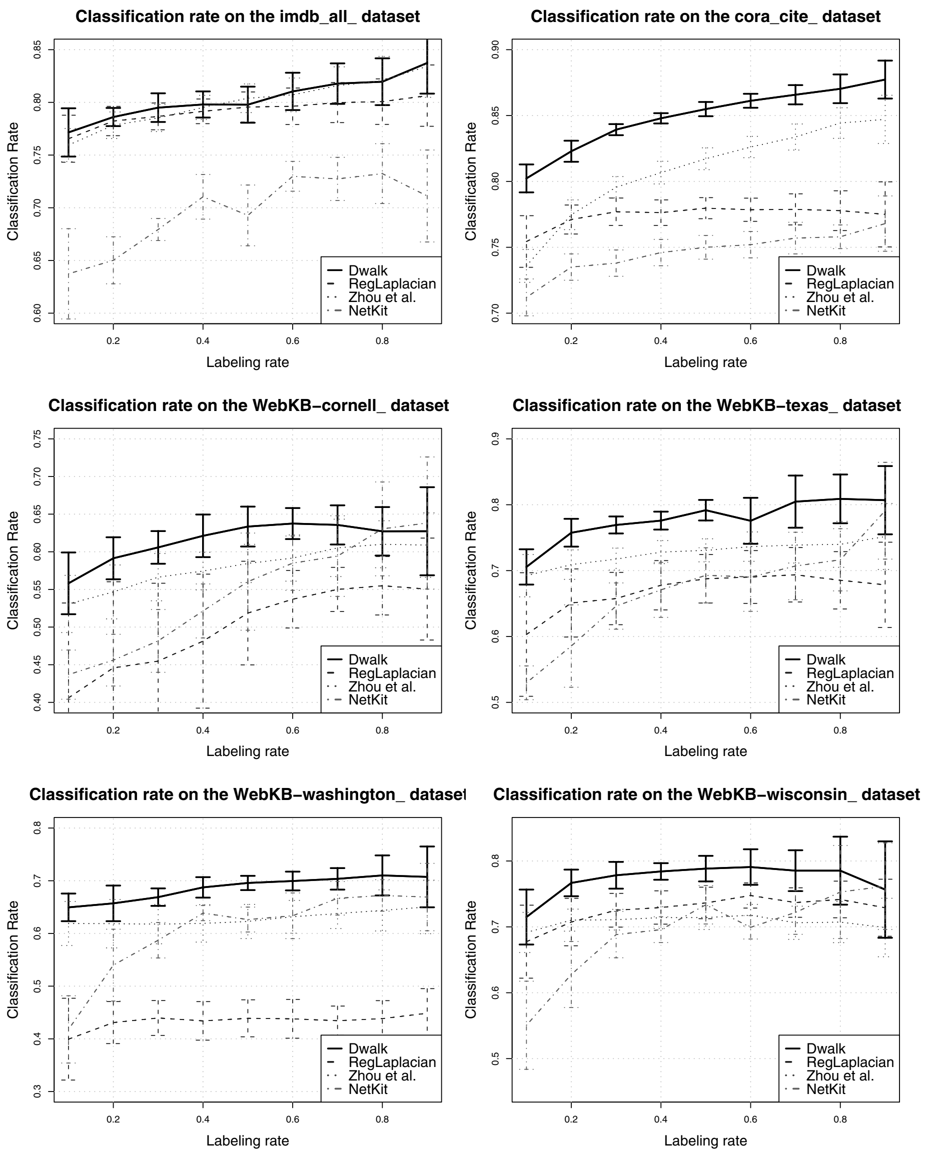

Figure 1. Classification rate of $\mathcal{D}$-walks and the three competing methods on each dataset. Error bars report standard deviations over 10 independent runs.

6.2 Methodology

Performances of the $\mathcal{D}$-walks approach and competing methods are reported in Figure 1 for the four datasets presented above. A network topology, along with some node labels, generally contains a considerable amount of useful information for unlabeled nodes classification. Only a small proportion of labeled nodes might be sufficient to predict the unlabeled set. For this reason, we have considered several labeling rates, *i.e.* proportion of nodes for which the label is known. The labels of remaining nodes are removed and used as test. For each considered labeling rate, 10 random deletions were performed on which performances are averaged.

Our approach has been compared to the following existing methods: NetKit ([5]; "NetKit"), Zhou and Schölkopf method ([15]; "Zhou et al.") and the regularized laplacian kernel ([2] [3]; "RegLaplacian"). Hyper-parameters of each approach were optimized using an internal 10-folds cross-validation. For the $\mathcal{D}$-walks approach, the unique parameter to tune is L. The best value for L is computed in the range $\{1, \ldots, 100\}$. Optimal L values typically fall between 6 and 30, showing the interest of bounding the walk length (see the right hand side of Figure 2). Zhou et al. and the regularized laplacian kernel also require the regularization parameter α to be tuned between 0 and 1. Concerning NetKit, testing all the module configurations would have been very time-consuming. We chose here the parameters that generally provide good results [5]. More precisely, the local classifier inducer uses the class prior, the relational classifier inducer uses the weighted vote Relational Neighbor classifier and a relaxation labeling is used for the collective inferencing.

6.3 Results

This section details the classification accuracy of each method over the various datasets. Figure 1 reports classification rates on test data obtained by each approach as a function of the labeling rate.

As could be expected, the classification rate generally improves with higher labeling rate. We can clearly observe that the $\mathcal{D}$-walks approach outperforms competing approaches most noticeably when fewer labeled nodes are considered. Moreover, the variance of this method is generally significantly lower. $\mathcal{D}$-walks is also the fastest method. On data sets like `CORA` with 3583 nodes, it requires less than a second of CPU on a standard PC including the tuning of its hyper-parameter L with cross-validation. NetKit takes about 4.5 seconds and Zhou et al. approach requires about one minute when explicitly computing the matrix inverse.

The scalability of the $\mathcal{D}$-walks approach was assessed on much larger data sets which were artificially generated, up to 10 million edges. These random graphs were generated using the algorithm presented in [13]. This technique allows one to generate an undirected and unweighted graph with a prescribed degree sequence drawn from a power law. The parameters of the power law were tuned such that the mean degree equals 10. The computing times on a standard PC[7] for increasing graph sizes and L values are shown in Figure 2. Memory and time requirements are strongly limiting factors to apply the other approaches on such large graphs.

[7] Intel Core 2 Duo 2.4Ghz.

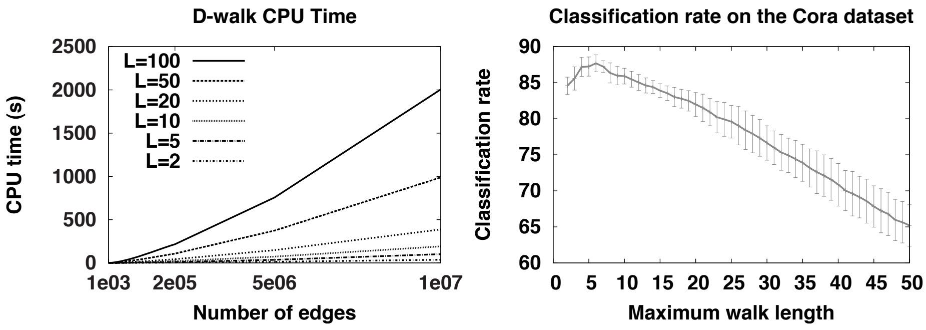

Figure 2. On the left, the CPU time of the $\mathcal{D}$-walks approach on artificially generated graphs of growing sizes, and for increasing maximum walk lengths L. The right side represents the classification rate on the `Cora` dataset, with a 0.9 labeling rate, when increasing the maximum walk length L. In this setting, the interest of bounding the walk length is clearly observed.

7 Conclusion

We propose in this paper a novel approach to the semi-supervised classification of nodes in a graph. The input graph is interpreted as a Markov chain (MC) in which random walks are performed. Particular random walks, called $\mathcal{D}$-walks, are introduced to define a node betweenness measure. A $\mathcal{D}$-walk starts in any node of a specific class y and ends as soon as some node of the same class is reached for the first time. The betweenness of a node q with respect to a class y is defined as the average number of times q is visited during $\mathcal{D}$-walks. Our betweenness measure is then refined by considering walks up to a prescribed length. Bounding the walk length provides systematically a better classification rate with the additional benefit that the betweenness can be computed very efficiently using forward and backward recurrences. The classification of all the unlabeled nodes in the graph can be performed with a time complexity $\Theta(|\mathcal{Y}|.m.L)$ where $|\mathcal{Y}|$ is the number of classes, m is the number of edges in the input graph and L is the maximum walk length considered. Moreover, the memory requirement is $\Theta(m + L.n)$, allowing one to take benefit of the possible graph sparseness. Such low complexities enable to deal with very large graphs containing several millions of nodes and edges. An extension of this approach is proposed to incorporate descriptive node features (*i.e.* attributes attached to each node) during classification.

Experiments on real-life databases show that the $\mathcal{D}$-walks approach outperforms three state-of-the-art approaches. The benefit of using bounded walks is empirically observed as better classification rates are obtained with walk length typically bounded between 2 and 10. We also show experimentally that the proposed approach easily scales up to large-scale problems thanks to its linear time complexity.

Our future work includes several extensions of the proposed approach. The interest of incorporating node features, as described in section 4, should be assessed experimentally. A typical case study would be the labeling of protein-protein interaction network. The node features could include gene expression measurements coding for the corresponding proteins. The $\mathcal{D}$-walks method could be further extended to perform regression on a graph, that is to predict a continuous value on nodes rather than a class label. Finally, we would like to investigate the use of bounded random walks in item recommendation systems through collaborative filtering.

References

1. Callut, J.: First Passage Times Dynamics in Markov Models with Applications to HMM Induction, Sequence Classification, and Graph Mining. Phd thesis dissertation, Universite catholique de Louvain (October 2007)
2. Chebotarev, P., Shamis, E.: The matrix-forest theorem and measuring relations in small social groups. Automation and Remote Control 58(9), 1505–1514 (1997)
3. Chebotarev, P., Shamis, E.: On proximity measures for graph vertices. Automation and Remote Control 59(10), 1443–1459 (1998)
4. Kemeny, J.G., Snell, J.L.: Finite Markov Chains. Springer, Heidelberg (1983)
5. Macskassy, S.A., Provost, F.: Classi cation in networked data: A toolkit and a univariate case study. J. Mach. Learn. Res. 8, 935–983 (2007)
6. Newman, M.E.J.: A measure of betweenness centrality based on random walks. Social networks 27, 39–54 (2005)
7. Norris, J.R.: Markov Chains. Cambridge University Press, United Kingdom (1997)
8. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report, Computer System Laboratory, Stanford University (1998)
9. Rabiner, L., Juang, B.-H.: Fundamentals of Speech Recognition. Prentice-Hall, Englewood Cliffs (1993)
10. Smola, A.J., Kondor, R.: Kernels and regularization on graphs. In: Schölkopf, B., Warmuth, M.K. (eds.) COLT/Kernel 2003. LNCS (LNAI), vol. 2777, pp. 144–158. Springer, Heidelberg (2003)
11. Szummer, M., Jaakkola, T.: Partially labeled classification with markov random walks. In: Advances in Neural Information Processing Systems, vol. 14, pp. 945–952 (2002)
12. Tsuda, K., Noble, W.S.: Learning kernels from biological networks by maximizing entropy. Bioinformatics 20(1), 326–333 (2004)
13. Viger, F., Latapy, M.: Efficient and simple generation of random simple connected graphs with prescribed degree sequence. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 440–449. Springer, Heidelberg (2005)
14. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: ICML 2005: Proceedings of the 22nd international conference on Machine learning, pp. 1036–1043. ACM, New York (2005)
15. Zhou, D., Schölkopf, B.: Learning from labeled and unlabeled data using random walks. In: Rasmussen, C.E., Bülthoff, H.H., Schölkopf, B., Giese, M.A. (eds.) DAGM 2004. LNCS, vol. 3175, pp. 237–244. Springer, Heidelberg (2004)