The Chinese University of Hong Kong
Department of Computer Science and Engineering

Ph.D. – Term Paper

Title:	Heat Diffusion Model and its Applications
Name:	Yang Haixuan
Student I.D.:	03499020
Contact Tel. No.:	6201-4825
Email A/C:	hxyang@cse.cuhk.edu.hk
Supervisor:	Prof. Irwin King & Prof. Michael R. Lyu
Markers:	Prof. Christopher C. Yang (SEEM) & Prof. Evangeline F.Y. Young
Mode of Study:	Full-time
Submission Date:	November 30, 2005
Term:	4
Fields:	
Presentation Date:	December 2, 2005
Time:	2:15–3:00 pm (to be confirmed)
Venue:	Rm. 1027, Ho Sin-Hang Engineering Building

Heat Diffusion Model and its Applications

Abstract

We establish a heat diffusion model on a graph by imitating the way that heat flows in a medium with a geometric structure, and we apply the heat diffusion model in classification and in similarity ranking on the Web Pages.

In application on classification, we propose two novel classification algorithms, Non-propagating Heat Diffusion Classifier (NHDC) and Propagating Heat Diffusion Classifier (PHDC). In NHDC, an unlabelled data is classified into the class that diffuses the most heat to the unlabelled data after one local diffusion from time 0 to a small time period, while in PHDC, an unlabelled data is classified into the class that diffuses the most heat to the unlabelled data in the propagating effect of the heat flow from time 0 to time t. In other words, we measure the similarity between an unlabelled data and a class by the heat amount that the unlabelled data receives from the set of labelled data in the class, and then classify the unlabelled data into the class with the most similarity. Unlike the traditional method, in which the heat kernel is applied to a kernel-based classifier we employ the heat kernel to construct the classifier directly; moreover, instead of imitating the way that the heat flows along a linear or nonlinear manifold, we let the heat flow along a graph formed by the k-nearest neighbors. An important and special feature in both NHDC and PHDC is that the kernel is not symmetric. We show theoretically that PWA (Parzen Window Approach when the window function is a multivariate normal kernel) and KNN are actually special cases of NHDC model, and that PHDC has the ability to approximate NHDC. Experiments show that NHDC performs better than PWA and KNN in prediction accuracy, and that PHDC performs better than NHDC.

In application on the Web pages, we propose a novel ranking algorithm called DiffusionRank, motivated by the way that heat flows, which reflects the complex relationship between nodes in a graph (or points on a geometry). Since the incomplete information about the Web structure causes inaccurate results of various ranking algorithms, we also propose a solution to this problem by formulating a new framework called, Predictive Random Graph Ranking, in which we generate a random graph based on the known information about the Web structure. The random graph can be considered as the predicted Web structure, on which ranking algorithm are expected to be improved in accuracy. For this purpose, we extend some current ranking algorithms from a static graph to a random graph. Experimental results show that the Predictive Random Graph Ranking framework can improve the accuracy of the ranking algorithms such as PageRank, Common Neighbor, and DiffusionRank.

Contents

Chapter 1

Introduction

Heat diffusion is a physical phenomena. In a medium, heat always flow from position with high temperature to position with low temperature. Heat kernel is used to describe the amount of heat that one point receives from another point.

Recently, the idea of heat kernel on a manifold is borrowed successfully in applications such as dimension reduction [1] and classification[15, 12]. In [15], the authors approximate the heat kernel for multinomial family in a closed form, from which great improvements are obtained over the use of Gaussian or linear kernels. In [12], the authors propose the use of discrete diffusion kernel to discrete or categorical data, and show that the simple diffusion kernel on the hypercube can result in good performance for such data. In [1], the authors employ heat kernel to construct weight of a neighborhood graph, and apply it to a non-linear dimensionality reduction algorithm.

Based on the successful applications of the heat kernel on the classification problem, it is natural to explore the use of heat kernel in a wider area where the underlying geometry is unknown or its heat kernel cannot be approximated in the same way as in [15]. To achieve our goal, we represent the underlying geometry by a finite neighborhood graph in the application on classification, instead of approximating the heat kernel in a given geometry; in application on the Web pages, the link structure is a well-defined graph. Then we establish a heat diffusion model based on this graph, instead of on the manifold.

All the proposed applications mentioned in this paper are established on a graph, and some of them will be established on a random graph. For our convenience, we first give some notations, and all common notations can be seen in the Appendix. Throughout the paper, all the graphs mentioned are directed graphs. We denote a static graph by $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{(v_i, v_j) \,|\, \text{there is an edge from } v_i \text{ to } v_j\}$ is the set of all edges. Let $I(v_i)$ and $I(v_j)$ denote the nodes that link to node v_i and v_j respectively, and $|I(v_i)|$, $|I(v_j)|$ means

the in-degree of the v_i and v_j respectively. The definition of a random graph is given below, and is denoted by $RG = (V, P)$.

Definition 1: A random graph $RG = (V, P = (p_{ij}))$ is defined as a graph with a vertex set V in which the edges are chosen independently, and for $1 \leq i, j \leq |V|$ the probability of (v_i, v_j) being an edge is exactly p_{ij}. The original definition of random graphs in [3], is slightly changed to consider the situation of directed graphs.

The remaining of the paper is organized as follows. In Chapter 2, we establish a heat diffusion model on the graph. In Chapter 3, we present *Non-propagating Heat Diffusion Classifier* (NHDC) and *Propagating Heat Diffusion Classifier* (PHDC). In Chapter 4, we propose the *DiffusionRank* algorithm. To improve the accuracy of the various ranking algorithms including *DiffusionRank*, in Chapter 5, we establish a framework *Predictive Random Graph Ranking*. In Chapter 6, we demonstrate the experimental results. Finally, in Chapter 7, we show the conclusions and future work.

Chapter 2

Heat Diffusion Model

2.1 Heat Diffusion Model on a Manifold

If the manifold is known, the heat flow throughout a geometric manifold with initial conditions can be described by the following second order differential equation:

$$\begin{cases} \frac{\partial f}{\partial t} - \Delta f &= 0, \\ f(x,0) &= f_0(x), \end{cases}$$

where $f(x,t)$ is the heat at location x at time t, beginning with an initial distribution of heat given by $f_0(x)$ at time zero, Δf is the *Laplace-Beltrami operator* on a function f. In local coordinates, Δf is given by

$$\Delta f = \frac{1}{\sqrt{\det g}} \sum_j \frac{\partial}{\partial x_j} \left(\sum_i g^{ij} \sqrt{\det g} \frac{\partial f}{\partial x_i} \right)$$

[15]. When the manifold is the familiar Euclidean Space, Δ is the Laplacian, and Δf is simplified as

$$\Delta f = \sum_i \frac{\partial^2 f}{\partial x_i^2}.$$

The heat or diffusion kernel $K_t(x,y)$ [15] is a special solution to the heat equation with a special initial condition called the delta function $\delta(x-y)$, which has the following properties: $\delta(x-y) = 0$ for $x \neq y$; $\int_{-\infty}^{+\infty} \delta(x-y)dx = 1$. The delta function $\delta(x-y)$ in the heat diffusion setting has the physical meaning – it describes a unit heat source at position y when there is no heat in other positions. Based on this, the heat kernel $K_t(x,y)$ describes the heat distribution at time t diffusing from the initial unit heat source at position y, and thus describes the connectivity (which is considered as a kind of similarity) between x and y.

Since arbitrary initial conditions can be considered as a combination of heat sources with different intensities at different positions, as a consequence of the linearity of the heat equation, the heat kernel can be used to generate

the solution to the heat equation according to the following equation

$$f(x,t) = \int_M K_t(x,y) f_0(y) dy.$$

The heat kernel $K_t(x,y)$ can be considered as a generalization of Gaussian density. This is because that when the underlying manifold is a flat n-dimensional Euclidean space, the heat kernel $K_t(x,y)$ has an explicit form

$$(4\pi t)^{-\frac{m}{2}} \exp(-\frac{||x-y||^2}{4t}), \tag{2.1}$$

which is the same as the Gaussian density. When the geometric manifold varies, the corresponding heat kernel varies and can be considered as the generalization of Gaussian density from flat Euclidean space to general manifold.

However, it is very difficult to find the manifold that the data or the Web pages lie in; even the manifold is known, it is very difficult to find the heat kernel $K_t(x,y)$, which involves solving Eq. (**??**) with the delta function as the initial condition. This motivates us to investigate the heat flow on a graph, and employ the heat diffusion behavior between nodes as the similarity measure between nodes.

2.2 Heat Diffusion Model on a Static Graph

On graph G, the edge (v_i, v_j) is considered as a pipe that connects to nodes v_i and v_j. The value $f_i(t)$ describes the heat at node v_i at time t, beginning from an initial distribution of heat given by $f_0(i)$ at time zero. $f(t)$ (f(0)) denotes the vector consisting of $f_i(t)$ ($f_0(i)$). We establish our model as follows. Suppose, at time t, each node v_i receives $RH(i,j,t,\Delta t)$ amount of heat from its antecedent v_j during a period of Δt. We have three assumptions:

1. The heat $RH(i,j,t,\Delta t)$ should be proportional to the time period Δt.
2. The heat $RH(i,j,t,\Delta t)$ should be proportional to the the heat at node v_j.
3. The heat $RH(i,j,t,\Delta t)$ is zero if there is no link from v_j to v_i.

As a result, v_i will receive $\sum_{j:(v_j,v_i)\in E} \sigma_j f_j(t) \Delta t$ amount of heat from all its antecedents.

On the other hand, node v_i diffuses $DH(i,t,\Delta t)$ amount of heat to its subsequent nodes. We assume that

1. The heat $DH(i,t,\Delta t)$ should be proportional to the time period Δt.
2. The heat $DH(i,t,\Delta t)$ should be proportional to the the heat at node v_i.
3. The heat $DH(i,t,\Delta t)$ is proportional to its out-degree $d^+(v_i)$.

As a result, node v_i will diffuse $\beta_i d^+(v_i) f_i(t)\Delta t$ amount of heat to all its subsequent nodes.

To sum up, the heat difference at node v_i between time $t + \Delta t$ and time t will be equal to the sum of the heat that it receives, deducted by what it diffuses. This is formulated as

$$f_i(t + \Delta t) - f_i(t) = -\beta_i d^+(v_i) f_i(t)\Delta t + \sum_{j:(v_j,v_i)\in E} \sigma_j f_j(t)\Delta t \tag{2.2}$$

For simplicity, we assume $\sigma_j = \sigma$, and $\beta_i = \beta$. To find a closed form solution to Eq. (2.2), we express it in a matrix form:

$$\frac{f(t + \Delta t) - f(t)}{\Delta t} = \sigma H f(t), \tag{2.3}$$

where $H = (H_{ij})$, and

$$H_{ij} = \begin{cases} -\frac{\beta}{\sigma} d^+(i), & j = i, \\ 1, & (v_j, v_i) \in E, \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

Let $t = 0$, Eq. (2.3) can be rewritten as

$$f(\Delta t) = (I + \sigma \Delta t H) f(0). \tag{2.5}$$

Assume $N = 1/\Delta t$ is an integer, then we have

$$\begin{aligned} f(1) &= f(N \cdot \Delta t) \\ &= (I + \sigma \Delta t H) f((N-1) \cdot \Delta t) \\ &= \dots \\ &= (I + \sigma \Delta t H)^N f(0). \end{aligned} \tag{2.6}$$

Eq. (2.6) is one closed form solution to Eq. (2.2) in the setting of discrete heat diffusion, where it describes the $N-$step heat distribution at a time period of Δt from time 0 to time 1. The matrix $(I + \sigma \Delta t H)^N$ is called as *Discrete Diffusion Kernel* in the sense that the heat diffusion process stops after a number of steps, and in each step, nodes diffuse their heat only to their subsequent nodes.

Next, we try to find another closed form solution to Eq. (2.2) in the setting of continuous heat diffusion. In the limit $\Delta t \to 0$, Eq. (2.3) becomes

$$\frac{d}{dt} f(t) = \sigma H f(t). \tag{2.7}$$

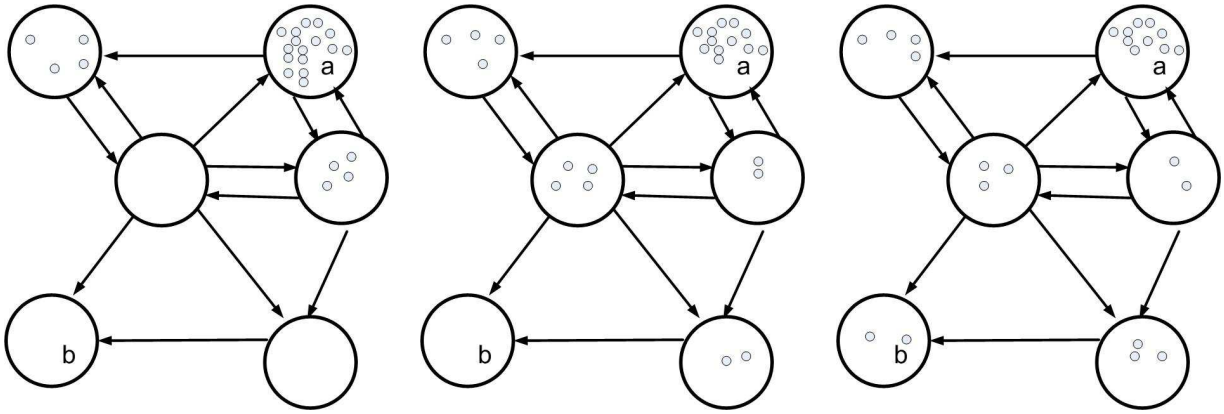
Solving Eq. (2.7), we get $f(t) = e^{\sigma t H} f(0)$, especially we have

$$f(1) = e^{\sigma H} f(0), \tag{2.8}$$

where $e^{\sigma H}$ is defined as

$$e^{\sigma H} = I + \sigma H + \frac{\sigma^2}{2!}H^2 + \frac{\sigma^3}{3!}H^3 + \cdots . \tag{2.9}$$

The matrix $e^{\sigma H}$ is called as *Continuous Diffusion Kernel* in the sense that the heat diffusion process continues infinitely many times after the nodes diffuse their heat to their subsequent nodes for the first time. In Figure 2.1,



Step 1 Step 2 Step 3

Figure 2.1. Illustration on Heat Diffusion

we illustrate the heat flow on a graph. There are six nodes in the graph, and there are links between them. We want to show how the heat diffuse from node a to node b. Initially there is heat only in node a, at Step 1, heat diffuse from a to its two subsequent nodes along the links, in Step 2, the heat diffuse further from nodes with high heat to nodes with low heat. At Step 3, node b receives heat from it two antecedents. The heat distribution in Step 3 can reflect the relationship between node a and other nodes, which is caused by the graph structure.

In case of weighted graphs, Eq. (2.2) can be changed to

$$f_i(t+\Delta t) - f_i(t) = \sum_{j:(j,i)\in E} \alpha \cdot \exp(-\frac{w_{ij}^2}{\beta})(f_j(t) - f_i(t))\Delta t \tag{2.10}$$

And the matrix $H = (H_{ij})$ is changed to be

$$H_{ij} = \begin{cases} -\sum_{k:(k,i)\in E} \exp(-\frac{w_{ik}^2}{\beta}), & j = i; \\ \exp(-\frac{w_{ij}^2}{\beta}), & (j,i) \in E; \\ 0, & \text{otherwise}. \end{cases} \tag{2.11}$$

Theorem 1: The solution in Eq. (2.8) has the property of heat preserving.

2.3 Heat Diffusion Model on a Random Graph

When the graph is uncertain, we need to establish heat diffusion model on a random graph.

On a random graph $RG = (V, P)$, where $P = (p_{ij})$ is the probability of the edge (v_i, v_j) exists. In such a random graph, the expected heat difference at node i between time $t + \Delta t$ and time t will be equal to the sum of the expected heat that it receives from all its antecedents, deducted by the expected heat that it diffuses. Since the probability of the link (v_j, v_i) is p_{ji}, we have

$$f_i(t + \Delta t) - f_i(t) = -\beta\, RD^+(v_i)\, f_i(t)\, \Delta t + \sum_{j:(v_j,v_i)\in E} \sigma p_{ji} f_j(t) \Delta t, \tag{2.12}$$

where $RD^+(v_i)$ is the expected out-degree of node v_i, it is defined as $\sum_k p_{ik}$. The Eq. (2.3) is changed accordingly

$$\frac{f(t + \Delta t) - f(t)}{\Delta t} = \sigma R f(t), \tag{2.13}$$

where $R = (R_{ij})$, and

$$R_{ij} = \begin{cases} -\frac{\beta}{\sigma} \sum_k p_{ik}, & j = i; \\ \\ p_{ji}, & j \neq i. \end{cases} \tag{2.14}$$

Assume $N = 1/\Delta t$ is an integer, then we have Eq. (2.6) is changed to be

$$f(1) = (I + \sigma \Delta t R)^N f(0). \tag{2.15}$$

Moreover, Eq. (2.8) is changed to be

$$f(1) = e^{\sigma R} f(0), \tag{2.16}$$

where $e^{\sigma R}$ is defined as

$$e^{\sigma R} = I + \sigma R + \frac{\sigma^2}{2!} R^2 + \frac{\sigma^3}{3!} R^3 + \cdots. \tag{2.17}$$

The matrix $(I + \sigma \Delta t R)^N$ in Eq. (2.15) and matrix $e^{\sigma R}$ in Eq. (2.16) are called *Discrete Diffusion Kernel* on the random graph and the *Continuous Diffusion Kernel* on the random graph respectively.

First we give our notation for the heat diffusion model on graph. Consider a directed weighted graph $G = (V, E, W)$, where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{(v_i, v_j) \,|\, \text{there is an edge}$
from v_i to $v_j\}$ is the set of all edges, and $W = (w_{ij})$ is the weight matrix. Different from the normal undirected weighed graph, the edge (v_i, v_j) is considered as a pipe that connects to nodes i and j, and the weight w_{ij} is considered as the length of the pipe (v_i, v_j). The value $f_i(t)$ describes the heat at node i at time t, beginning from an initial distribution of heat given by $f_0(i)$ at time zero.

Based on the two closed form solutions Eq. (2.5) and Eq. (2.8), we establish two different classifiers in the next two sections.

Chapter 3

Heat Diffusion Classifiers

3.1 Non-propagating Heat Diffusion Classifier

Assume that there are c classes, namely, $C_1, C_2, \ldots, C_c$. Let the labelled data set contains M samples, represented by $(\mathbf{x}_i, k_i)$ ($i = 1, 2, \ldots, M$), which means that the data point $\mathbf{x}_i$ belongs to class C_{k_i}. Suppose the labelled data set contains M_k points in class C_k so that $\sum_k M_k = M$. Let an unlabelled data set contains N unlabelled samples, represented by $\mathbf{x}_i$ ($i = M+1, M+2, \ldots, M+N$).

We first employ the neighborhood construction algorithm commonly used in the literature, for example in [1], [23], [21] and [22], to form a graph for all the data. Then we apply the non-propagating heat diffusion kernel to the graphs. For the purpose of classification, for each class C_k in turn, we set the initial heat at the labelled data in class C_k to be one and all other data to be zero, then calculate the amount of heat that each unlabelled data receives from the labelled data in class C_k. Finally, we assign the unlabelled data to the class from which it receives most heat. More specifically, we describe the resulting non-propagating Heat Diffusion-Based Classifier as follows.

[Step 1: Construct neighborhood graph] Define graph G over all data points both in the training data set and in the unlabelled data set by connecting points $\mathbf{x}_j$ and $\mathbf{x}_i$ from $\mathbf{x}_j$ to $\mathbf{x}_i$ if $\mathbf{x}_j$ is one of the K nearest neighbors of $\mathbf{x}_i$ measured by the Euclidean distance. Let $d(i,j)$ be the Euclidean distance between point $\mathbf{x}_i$ and point $\mathbf{x}_j$. Set edge weight w_{ij} equal to $d(i,j)$ if $\mathbf{x}_i$ is one of the K nearest neighbors of $\mathbf{x}_j$, and set $n = M + N$.

[Step 2: Compute the Non-propagating Heat Kernel] Using Eq. (2.14), get the Non-propagating Heat Kernel H.

[Step 3: Compute the Heat Distribution] Let

$$f^k(0) = (x_1^k, x_2^k, \ldots, x_M^k, \underbrace{0, 0, \ldots, 0}_{N})^T,$$

$k = 1, 2, \ldots, c$, where $x_i^k = 1$ if $C_{k_i} = C_k$, $x_i^k = 0$ otherwise. Then we obtain c results for $f(\Delta t)$, namely, $f^k(\Delta t) = Hf^k(0)$, $k = 1, 2, \ldots, c$.

By Eq. (2.5), $f^k(\Delta t)$ should be equal to $(I + \alpha \Delta t H) f^k(0)$, but the identity matrix I and the constant $\alpha \Delta t$ have no effect on the classifier introduced in Step 4, so we simply let $f^k(\Delta t) = Hf^k(0)$. $f^k(0)$ means that all the data points in class C_k have a unit heat at the initial time while other data points have no heat, and the corresponding result $f^k(\Delta t)$ means that the heat distribution at time Δt is caused by the initial heat distribution $f^k(0)$.

[Step 4: Classify the data] For $l = 1, 2, \ldots, N$, compare the p-th ($p = M+l$) components of $f^1(\Delta t), f^2(\Delta t), \ldots, f^c(\Delta t)$, and choose class C_k such that $f_p^k(\Delta t) = \max_{q=1}^c f_p^q(\Delta t)$, i.e., choose the class that distributes the most heat to the unlabelled data $\mathbf{x}_p$, then classify the unlabelled data $\mathbf{x}_p$ to class C_k.

In Figure 3.1, we illustrate a neighborhood graph, in which three cases are represented by circle and labelled as class 1, two cases are represented by square and labelled as class 2, and one case is represented by a triangle and is unlabelled. According to Step 1, there is an edge from $\mathbf{x}_j$ to $\mathbf{x}_i$ if $\mathbf{x}_j$ is one of the K nearest neighbors of $\mathbf{x}_i$, and hence the in-degree of each node is K. In the graph in Figure 3.1, K is set to be 2.

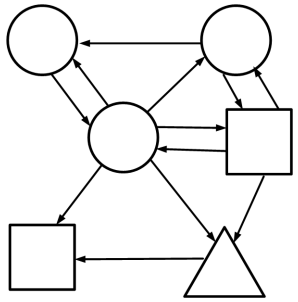


Figure 3.1. Neighborhood Graph

Figure 3.2 shows how heat flows from one node to another node when the initial heat is 1 at nodes in class 1 and 0 at other nodes. A node diffuses heat only to its successors through the directed edge. As a result of the non-propagating heat diffusion, one square receives heat, represented by two small circles, from its two circle predecessors; one square receives heat, represented by one small circle, from its one circle predecessor; the unlabelled data (triangle) receives heat, represented by one small circle, from its one circle predecessor.

Similarly Figure 3.3 shows the result of non-propagating heat flow when the initial heat is 1 at nodes in class 2 and 0 at other nodes.

The unlabelled data (triangle) receives heat both from nodes in class 1 and nodes in class 2. According to Step 4, we classify the unlabelled data as the class from which it receives the most heat. Through comparison the amount of heat in the triangle in Figure 3.2 and Figure 3.3, we classify the unlabelled data to class 2.

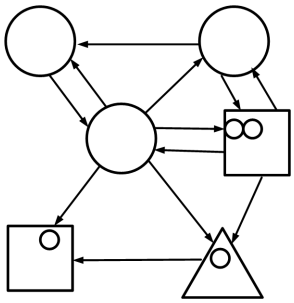


Figure 3.2. Non-propagating Heat Diffusion Result on the Neighborhood Graph

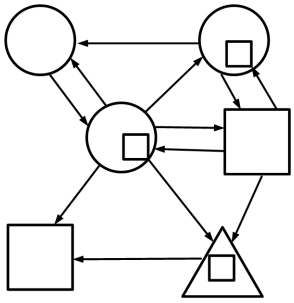


Figure 3.3. Non-propagating Heat Diffusion Result on the Neighborhood Graph

In this non-propagating heat diffusion classifier (NHDC), we only consider the heat flow in a small time period, and heat diffuses only once during such a period. We have two free parameters in NHDC: K and β. In the next section, we consider the propagating effect of infinitely many times of heat flow: The heat diffuses to its neighbors first, then these neighbors diffuse the heat further to their own neighbors. This process continues until an appropriate time t is reached.

3.2 Propagating Heat Diffusion Classifier

In this classifier, we replace the non-propagating heat diffusion kernel H with the propagating heat diffusion kernel $e^{\gamma H}$. Consequently, the algorithm in Section 3.1 changes to the following.

[Step 1: Construct neighborhood graph] The same as Step 1 in Section 3.1.

[Step 2: Compute the Propagating Heat Kernel] Using Eq. (2.14) and Eq. (2.9), get the Heat Kernel $e^{\gamma H}$.

[Step 3: Compute the Heat Distribution] $f^k(0)$ is the same as Step 3 in Section 3.1. Using Eq. (2.8), we obtain c results for $f(t)$, namely, $f^k(t) = e^{\gamma H} f^k(0)$, $k = 1, 2, \ldots, c$.

[Step 4: Classify the data] For $l = 1, 2, \ldots, N$, compare the p-th ($p = M+l$) components of $f^1(t), f^2(t), \ldots, f^c(t)$, and choose class C_k such that $f_p^k(t) = \max_{q=1}^{c} f_p^q(t)$, i.e., choose the class that distributes the most heat to the unlabelled data $\mathbf{x}_p$ from time 0 to time t, then classify the unlabelled data $\mathbf{x}_p$ to class C_k.

Since we consider the propagating effect of heat diffusion, this classifier is called Propagating Heat Diffusion Classifier (PHDC). We have three free parameters in AHDBC: K, β and γ.

Different from NHDC, after the first heat diffusion, the heat will continue to diffuse in PHDC. The second heat diffusion is based on the result of the first diffusion, which is roughly illustrated by Figure 3.4 and Figure 3.5. The tiny circles mean less amount of heat transmitted in the second diffusion, which may directly come from data (circle) in class 1 or indirectly from data (square) in class 2. The tiny squares have similar meaning. For example, there are two tiny circles in the left-lowest large square. They are the results of the second diffusion: One tiny circle is transmitted indirectly from the small circle in the right large triangle, and the other tiny circle is directly from the large circle in the middle. When the time period Δt tends to zero and in fact our model acts this way, there is infinitely many times $t/\Delta t$ of heat diffusion from time 0 to time t.

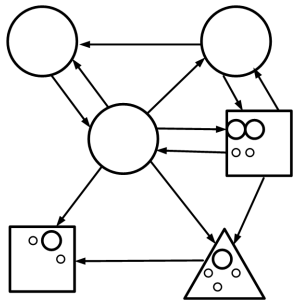


Figure 3.4. Second Heat Diffusion Result on the Neighborhood Graph

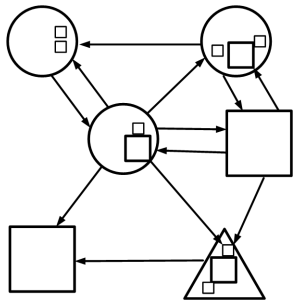


Figure 3.5. Second Heat Diffusion Result on the Neighborhood Graph

Remark In Step 1, we construct only one graph over both labelled data and unlabelled data by the method of K nearest neighbors. There are many variants in this step:

1. We can construct the graph by other methods such as ϵ-neighborhood.

2. We can construct c graphs: For each class C_k in turn, construct graph by connecting all the unlabelled data points and data points with label k. In such case, Step 3 and Step 4 need to be changed correspondingly.

3.3 Interpretation

In Section 2, we assume that the heat diffuses in the pipe in the same way as it does in the m-dimensional Euclidean space. Next we will justify this assumption.

It turns out [1] that in an appropriate coordinate system $K_t(x, y)$ on a manifold is approximately the Gaussian:

$$K_t(x,y) = (4\pi t)^{-\frac{m}{2}} \exp(-\frac{||x-y||^2}{4t})(\phi(x,y) + O(t)),$$

where $\phi(x, y)$ is a smooth function with $\phi(x, x) = 1$ and $O(t)$ represents an ignorable term when t is small. Therefore when x and y are close and t is small, we have

$$K_t(x,y) \approx (4\pi t)^{-\frac{m}{2}} \exp(-\frac{||x-y||^2}{4t}).$$

For more details, see [1] and [20].

In our graph heat diffusion model in Section 2, we first consider the heat flow in a small time period Δt, and the pipe length between node i and node j is small (recall that only when j is one of the K nearest neighbors, we create an edge from j to i). So the above approximation can be used in our model, and we rewrite it as follows:

$$K_{\Delta t}(i,j) \approx (4\pi\Delta t)^{-\frac{m}{2}} \exp(-\frac{w_{ij}^2}{4\Delta t}). \tag{3.1}$$

According to the Mean-Value Theorem and the fact that $K_0(i, j) = 0$, we have

$$\begin{aligned} K_{\Delta t}(i,j) &= K_{\Delta t}(i,j) - K_0(i,j) \\ &= \frac{dK_{\Delta t}(i,j)}{d\Delta t}\Big|_{\Delta t=\beta} \Delta t \\ &\approx \alpha \cdot \exp(-\tfrac{w_{ij}^2}{4\beta})\Delta t, \end{aligned}$$

where the last approximation is based on Eq. (3.1), β is a parameter that depends on Δt, and $\alpha = \frac{1}{4}w_{ij}^2\beta^{-m/2-2} - \frac{1}{2}m\beta^{-m/2-1}$. To make our model concise, α and β simply serve as free parameters that unrelated to Δt and w_{ij}. This explains why we assume that the at time t, each node i receives $M(i, j, t, \Delta t) = \alpha \cdot \exp(-\frac{w_{ij}^2}{\beta})(f_j(t) - f_i(t))\Delta t$ amount of heat from its neighbor j.

3.4 Connections with Other Models and Related Work

In this section, we establish connections between NHDC and other models, and connection between NHDC and PHDC. We show that PWA (Parzen Window Approach [2] when the window function is a multivariate normal

kernel) and KNN (K-Nearest-Neighbors) are actually special cases of NHDC, and that PHDC can approximate NHDC. Finally, we compare our heat kernel with those in the related work.

3.4.1 NHDC and Parzen Window Approach

First we review the Parzen Windows non-parametric method for density estimation, using Gaussian kernels. When the kernel function $H(u)$ is a multivariate normal kernel, a common choice for the window function, the estimate of the density at the point x is

$$\widetilde{p}(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{(2\pi h^2)^{d/2}}\exp(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}). \tag{3.2}$$

When applying it for classification, we need to construct the classifier through the use of Bayes's theorem. This involves modelling the class-conditional densities for each class separately, and then combining them with priors to give models for the posterior probabilities which can then be engaged to make classification decisions [2]. The class-conditional densities for class C_k can be obtained by extending Eq. (3.2):

$$\widetilde{p}(\mathbf{x}|C_k) = \frac{1}{M_k}\sum_{i:C_{k_i}=C_k}\frac{1}{(2\pi h^2)^{d/2}}\exp(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}), \tag{3.3}$$

while the priors can be estimated using $\widetilde{p}(C_k) = \frac{M_k}{M}$. Using Bayes' theorem, we get

$$\widetilde{p}(C_k|\mathbf{x}) = \frac{1}{Mp(\mathbf{x})(2\pi h^2)^{d/2}}\sum_{i:C_{k_i}=C_k}\exp(-\frac{||\mathbf{x}-\mathbf{x}_i||^2}{2h^2}). \tag{3.4}$$

If $K = n-1$, then the graph constructed in Step 1 will be a complete graph, and the matrix H in Eq. (2.14) becomes

$$H_{ij} = \begin{cases} -\sum_{k\neq i}\exp(-\frac{w_{ik}^2}{\beta}), & j = i; \\ \exp(-\frac{w_{ij}^2}{\beta}), & j \neq i. \end{cases} \tag{3.5}$$

Then, in NHDC, the heat $f_p^k(\Delta t)$ that unlabelled data $\mathbf{x}_p$ receives from the data points in class C_k will be equal to $\sum_{i:C_{k_i}=C_k}\exp(-||\mathbf{x}_p-\mathbf{x}_i||^2/\beta)$, which is the Eq. (3.4) if we let $\gamma = 1/Mp(\mathbf{x})(2\pi h^2)^{d/2}$, and $\beta = 2h^2$. This means that Parzen Window Approach when the window function is a multivariate normal kernel can be considered as a special case of NHDC (when we let $K = n-1$ in NHDC).

3.4.2 NHDC and KNN

If β tends to infinity, then $\exp(-\frac{w_{ij}^2}{\beta})$ will tend to one, and the matrix H in Eq. (2.14) becomes

$$H_{ij} = \begin{cases} -K_i, & j = i; \\ 1, & \mathbf{x}_j \text{ is one of the } K \text{ nearest neighbors of } \mathbf{x}_i; \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

Here K_i is the outdegree of the point $\mathbf{x}_i$ (note that the indegree of the point $\mathbf{x}_i$ is K). Then, in NHDC, the heat $f_p^q(\Delta t)$ that unlabelled data x_p receives from the data points in class C_q will be equal to

$$f_p^q(\Delta t) = \sum_{i:l_i=C_q} 1 \ = K_q,$$

where K_q is the number of the labelled data points from class C_q, which are the K nearest neighbors of the unlabelled data point $\mathbf{x}_p$. Note that when $N = 1$, i.e., when the number of unlabelled data is equal to one, $\sum_{q=1}^{c} K_q = K$. According to Step 4, we will classify the unlabelled data $\mathbf{x}_p$ to the class C_k such that $f_p^k(\Delta t) = K_k$ is the maximal among all $f_p^q(\Delta t) = K_q$. This is exactly what KNN does, and so KNN can be considered as a special case of NHDC (when β tends to infinity and $N = 1$).

3.4.3 NHDC and PHDC

When the parameter γ is small, we can approximate $e^{\gamma H}$ in Eq. (2.9) by its first two items, i.e.,

$$e^{\gamma H} \approx I + \gamma H, \tag{3.7}$$

then in PHDC, $f^k(t) = e^{\gamma H} f^k(0) \approx f^k(0) + \gamma H f^k(0)$. As the constant γ and the first item $f^k(0)$ impose no effect on the classifier, PHDC possesses a similar classification ability in this case as NHDC, in which $f^k(\Delta t) = H f^k(0)$. This denotes the relation between NHDC and PHDC.

3.4.4 Related Work

The success in [15] is achieved partly because of the speciality of the geometry in the problem. For most geometries, however, there is no closed form solution for the heat kernel. Even worse, in most cases, the underlying geometry structure is unknown. In such cases, it is impossible to construct the heat kernel for the geometry in a closed form. In contrast, there is always a closed form solution – a heat kernel for the graph that approximates the geometry in our model. In [15] and [12], heat kernel is applied to a large margin classifier; in contrast, our kernel is employed directly to construct a classifier.

It is worthy to make a theoretical comparison between the heat kernel in our model and that in [12] because it is impossible to make an empirical comparison between them (as shown below in the second item, their applications are different), and because our heat kernel shows the same appearance $e^{\gamma H}$ as that in [12]. We list below the major differences between them:

1. When the graph is symmetric and β tends to infinity, the matrix H and the heat kernel $e^{\gamma H}$ in our model take the same form as that in [12].

2. Our classifier is mainly concerned with the real-valued data, while the proposed classifier in [12] aims at categorical data in their experiments.

3. Our graph is constructed by the K nearest neighbors in order to approximate the discrete structure of the unknown manifold, while in [12], for each attribute, a graph is constructed by a hypercube, and then the final diffusion kernel is the product of each individual diffusion kernel.

4. Our model is created by the imitation of the non-propagating heat diffusion and the propagating effect of the local heat diffusion. The heat flow in the pipe behaves in the way of locality, and thus it can approximate the heat kernel in the Euclidean space because the time period and the pipe length are small. However, in [12], there is no such consideration.

5. Limited to narrow applications, the kernel in [12] must satisfy two mathematical requirements to be able to serve as a kernel: It must be symmetric and positive semi-definite. In contrast, without the limitation of being applied to a kernel-based classifier, our heat kernel is not necessarily symmetric and positive semi-definite.

Nevertheless, it is interesting to combine these two models by considering the cases when there are both continuous attributes and categorical attributes in the data set. Besides, it is a challenge to apply our heat kernel to a kernel-based classifier when the kernel is not symmetric. These deserve further investigations, but are outside the scope of this paper.

Chapter 4

Heat Diffusion Ranking

In this section, we propose a new ranking model called *DiffusionRank*. The intuition is that all the Web pages in the World Wide Web are imagined to be drawn from a manifold. On the manifold, the heat flows from one point to another point, and in a given time period, if one point x receives much heat from another point y, we can say x and y are connected well, and thus x and y have a high similarity in the meaning of a high mutual connection.

We simulate the heat flow on a manifold by the heat flow on a graph since the World Wide Web is so complex that we cannot model it as a regular geometry with a known dimension. In this paper, the Web pages are considered to be drawn from an unknown manifold, and the link structure forms a directed graph, which is considered as an approximation to the unknown manifold. The heat kernel established on the Web graph is considered as the representation of relationship between Web pages. According to the Heat Kernel proposed in Section 2.2 and Section 2.3, we describe *DiffusionRank* as follows.

For a random graph, the matrix $(I + \sigma\Delta tR)^N$ or $e^{\sigma R}$ can measure the similarity relationship between nodes. Let $f_i(0) = 1, f_j(0) = 0$ if $j \neq i$, then the vector $f(0)$ represent the unit heat at node v_i while all other nodes has zero heat. For such $f(0)$ in a random graph, we can find the heat distribution at time 1 by using Eq. (2.15) or Eq. (2.16). The heat distribution is exactly the $i-$th row of the matrix of $(I + \sigma\Delta tR)^N$ or $e^{\sigma R}$. So the i-row j-column element h_{ij} in the matrix $(I + \sigma\Delta tR)^N$ or $e^{\sigma R}$ means the amount of heat that v_i can receive from v_j from time 0 to 1. Thus the value h_{ij} can be used to measure the similarity from v_j to v_i.

For a static graph, similarly the matrix $(I + \sigma\Delta tH)^N$ or $e^{\sigma H}$ can measure the similarity relationship between nodes.

The intuition behind is that the amount $h(i, j)$ of heat that a page v_i receives from a unit heat in a page v_j in a unit time embodies the extent of the link connections from page v_j to page v_i. Roughly speaking, when there are more paths from v_j to v_i, v_i will receive more heat from v_j, on the other hand, when the path length from v_j to v_i

is shorter, v_i will receive more heat from v_j. The final heat that v_i receives will depend on various paths from v_j to v_i.

The four advantages for *DiffusionRank* are shown below.

First, its solution has two forms, both of which are closed form. One takes the discrete form, and has the advantage of fast computing while the other takes the continuous form, and has the advantage of being analyzed theoretically.

Second, its solution is not symmetric, which better models the nature of relativity of similarity. For example, that one page links to an important page does not mean that this page is also important unless it is linked by the important page.

Third, it can be naturally employed to detect group-group relation. For example, if group 2 contains pages $(j_1, j_2, \ldots, j_s)$ group 1 contains pages $(i_1, i_2, \ldots, i_t)$, then the sum $\sum_{u,v} h_{i_u,j_v}$ has the meaning of total heat that group 1 receives from group 2, where h_{i_u,j_v} is the i_u-row j_v-column element of the heat kernel.

Fourth, it can be used to anti-manipulation. Let group 2 contains trusted Web pages $(j_1, j_2, \ldots, j_s)$, then for each page i, $\sum_v h_{i,j_v}$ is the heat that page i receives from the group 2, and can be computed by Eq. (2.6) in case of a static graph or Eq. (2.15) in case of a random graph, in which $f(0)$ is set to be a special initial heat distribution so that the trusted Web pages have unit heat while all the others have zero heat. In doing so, manipulated Web page will get a lower rank unless it has strong in-links from the trusted Web pages directly or indirectly. For such application of *DiffusionRank*, the computation complexity for *Discrete Diffusion Kernel* is the same as that for *PageRank* in cases of both a static graph and a random graph. This can be seen in Eq. (2.6) and Eq. (2.15), by which we need N iterations and for each iteration we need a multiplication operation between a matrix and a vector, while in Eq. (5.1) and Eq. (5.2) we also need a multiplication operation between a matrix and a vector for each iteration.

Chapter 5

Predictive Random Graph Ranking on the Web

While the *PageRank* algorithm [19] has proven to be very effective for ranking Web pages, inaccurate *PageRank* results are induced because of the incomplete information about the Web structure. This problem is caused by the following phenomena.

1. *The Web is Dynamic (temporal dimension)*–The link structure evolves temporally. Some links are created and modified, while others are destroyed.

2. *The Observer is Partial (spatial dimension)*–For different observers (or crawlers), the Web structure may be different.

3. *Links are Different (local dimension)*–Not all out-links are created equal. Some out-links are more significant than others. For example, some people may tend to put the most important link on the top of their pages.

For the problem of the incompleteness and impreciseness of the Web structure, we establish *Predictive Random Graph Ranking* framework. As illustrated in Figure 5, the framework consists of two stages:

- **Random Graph Generation Stage**–The first stage engages the temporal, spatial and local link information to construct a random graph that can better model the Web. Statistical and other methods can be applied to generate this random graph that can better approximate the incomplete Web.

- **Random Graph Ranking Stage**–The second stage takes the random graph output and then calculates the

ranking result based on a candidate ranking algorithm, such as, *PageRank*, *Common Neighbors*, *Jaccard's Coefficient*, *SimRank*, etc.

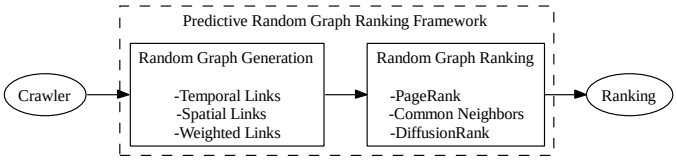


Figure 5.1. The Predictive Random Graph Ranking Framework.

The intuition in the *Predictive Random Graph Ranking* framework is that: the more accurately we know the structure of the Web, the more accurately we can infer about the Web.

DiffusionRank is another candidate in the *Predictive Random Graph Ranking* framework.

5.1 Some Standard Ranking Algorithms

We classify ranking techniques into two types: *Absolute Ranking* and *Relative Ranking*. *Absolute Ranking* assigns a real number to each page, and thus gives a total order for all pages. *PageRank* [19] belongs to *Absolute Ranking*. *Relative Ranking* assigns a real number to each pair of pages, and thus, for each one given page, determines a total order relative to the given page. *Common Neighbors* [18], *Jaccard's Coefficient* [16], and *SimRank* [9] belong to *Relative Ranking*.

5.1.1 Absolute Ranking

As a kind of *Absolute Ranking*, *PageRank* [19] gives the importance rank of Web page based on the link structure of the Web. The intuition behind *PageRank* is that it uses information external to the Web pages themselves–their in-links, and that in-links from "important" pages are more significant than in-links from average pages. Formally presented in [4], the Web is modelled by a directed graph $G = (V, E)$ in the *PageRank* algorithms, and the rank or "importance" x_i for page $v_i \in V$ is defined recursively in terms of pages which point to it:

$$x_i = \sum_{(j,i)\in E} a_{ij} x_j, \tag{5.1}$$

where a_{ij} is assumed to be $1/d_j$, d_j is the out-degree of page j. Or in matrix terms, $x = Ax$. When the concept of "random jump" is introduced, the matrix form in Eq. (5.1) is changed to

$$x = [(1-\alpha)ge^T + \alpha A]x, \tag{5.2}$$

where the parameter α is the probability of following the actual link from a page, $(1 - \alpha)$ is the probability of taking a "random jump", and g is a stochastic vector (i.e. $e^T g = 1$). Typically, $\alpha = 0.85$ and e is the vector of all ones.

5.1.2 Relative Ranking

In [16], the authors survey an array of methods for *Relative Ranking*, including *Common Neighbors*, *Jaccard's Coefficient*, and *SimRank*. All the methods assign a connection weigh $s(i, j)$ to pairs of nodes v_i and v_j, based on the input graph. The development of similarity search algorithms is motivated by the "related pages" queries of Web search engines and Web document classification [6]. Both applications require a similarity measure, which is computed by either the textual content of pages or the hyperlink structure or both. As in previous work [6, 9, 8], we focus on similarities solely determined by hyperlink structure of the Web graph.

Common Neighbors

Common neighbor model is based on the idea that two pages are more similar if they have more common neighbors. The common neighbors of v_i and v_j can be defined as $s(i, j) = |I(v_i) \cap I(v_j)|$. It means that if more nodes points to v_i and v_j at the same time, v_i and v_j are more similar. In [18], the author computes the probability of collaboration between scientists in the Los Alamos as a function of the times of their past collaboration. A pair of scientists with more previous collaborators are more likely to collaborate than those with less previous collaborators. In [16], the authors employ common neighbors to predict if any two authors will coauthor papers in the future.

Jaccard's Coefficient

Another commonly used similarity metric is the *Jaccard coefficient*, which is used to measure the probability that both v_i and v_j share a feature. In [16], the authors take features to be neighbors in graph, which corresponds to the measure $s(i, j) = |I(v_i) \cap I(v_j)|/|I(v_i) \cup I(v_j)|$. In this paper we utilize this approach as well to measure the similarity between two pages in the Web.

SimRank

SimRank is introduced in [9] to formalize the intuition that "two pages are similar if they are referenced by similar pages." Numerically this is specified by defining the *SimRank* score $s(i, j)$ of two pages v_i and v_j as the fixed

point of the following recursive definition,

$$s(i,j) = \begin{cases} 1, & i = j, \\ 0, & |I(v_i)||I(v_j)| = 0, i \neq j, \\ K \sum\limits_{u \in I(v_i), v \in I(v_j)} s(u,v), & \text{others}, \end{cases}$$

for some constant decay factor $C \in (0,1)$, where $K = \frac{C}{|I(v_i)||I(v_j)|}$. The *SimRank* iteration starts with $s(i,j) = 1$ for $i = j$ and $s(i,j) = 0$ otherwise.

5.2 Predictive Strategy

In this section, we first show the origin of the idea of the predictive strategy, then we show that the concept of a random graph is necessary, next we show how a random graph can be generated in various situations. This forms the first stage of the framework *Predictive Random Graph Ranking*. We continue to extend several ranking models from static graphs to random graphs. These are the second stage of the *Predictive Random Graph Ranking* framework .

5.2.1 Origin of Predictive Strategy

In [24], the authors propose a predictive ranking technique to improve the accuracy of *PageRank* through the estimation of the incomplete information caused by partial crawling on the Web. The more accurately estimated Web structure leads to a more accurate *PageRank* result. In this paper, we extend the basic idea in [24] from *PageRank* to a collection of ranking algorithms, from temporal incomplete information to spatial uncertainty and weighted links.

5.2.2 From Static Graphs to Random Graphs

The concept of a random graph is necessary for *PageRank*. For example, the graph in Figure 5.2 may be encountered by a crawler in the early stage if all the unvisited nodes are ignored. If we employ Eq. (5.1) and use the power iterative method to solve the page rank problem, then we will suffer the problem of divergence unless the entire initial values of x_i $(i = 1, 2, 3)$ take the value of $1/3$, which usually can not be found in practice. However, if we employ Eq. (5.2), the power iterative method will converge. This is because the modified matrix in Eq. (5.2) is a positive stochastic matrix, and so 1 is its largest absolute eigenvalue and no other eigenvalue whose absolute value is equal to 1, which is guaranteed by the Perron Theorem [17]. Behind Eq. (5.2), we can see that the Web

graph has been modelled as a random graph, in which, the original link exists with a probability of α, and there is a link that connects each pair of pages with a probability of $1 - \alpha$.

Furthermore, in the following, we discuss three situations: (1) *temporal links*, (2) *spatial links* and (3) *weighted links*, in which the concept of a random graph is also necessary.

Random Graph Generated Temporal Links

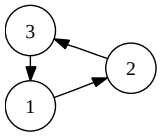


Figure 5.2. A static graph.

If we want to model the estimation about the temporal links, the concept of a random is necessary. In Figure 5.2, when the time continues, the crawler will visit more nodes, but at current time, the links (called temporal links) from the currently unvisited node are unknown. In general, it is difficult to estimate the temporal link structure accurately; however, some elementary estimation is possible. In this paper, we only estimate the in-degree of each node in the set of nodes that have been found, and thus some information about the link structure can be inferred statistically. For more discussions, see Section 5.2.3.

Random Graph Generated by Several Graphs

Several crawlers may visit some pages at different times and from different starting sites, and a link may exist for one crawler, but disappear for another. This causes the partial observer problem–the web graph is viewed differently from different points. Suppose that different Web graphs $G_i = (V_i, E_i), (i = 1, 2, \ldots, N)$ are obtained by N different observers (or crawlers). We can combine these different graphs and generate a random graph $RG = (V, P)$, where $V = \cup_{i=1}^{N} V_i, P = (p_{ij}), p_{ij} = n(i, j)/N, n(i, j)$ is the number of the graphs where the link (i, j) appears. The intuition behind is that the more a link is reliable, the more times different observers will find it.

Random Graph Generated by Weighted Links

We have observed that some out-links are more significant than others. As an example, we may model the out-link significance by the exponential decay rule: e^{1-k} where k is the out-link order number from a particular page. Then a random graph generated by this rule will be $P = (p_{ij})$ where $p_{ij} = 0$ if there is no link from i to j, and

$p_{ij} = e^{1-k(i,j)}$ if j is the $k(i,j)$-th out-link from i. By doing so, the significance of different out-links from a particular page is distinguished. The original static graph is changed to a random graph.

In next subsection, we emphasis on the problem of dangling nodes, which is caused by the nature of the dynamic Web. This problem is handled by predicting the link structure as a random graph. To sum up, it is necessary to extend the current ranking algorithms from a static graph to a random graph.

5.2.3 From Visited Nodes to Dangling Nodes

Why We Consider Dangling Nodes

Pages that either have no out-link or have no known out-link are called dangling nodes [4]. In [19], the authors suggested simply removing the pages without out-link and the links pointing to them. After doing so, it is suggested that they can be "added back in" without significantly affecting the results. However the situation is changed now and the dangling nodes problem has to be handled more accurately and directly:

On the one hand, we can see that the *PageRank* algorithm depends on part of the Web structure, and that the visited fraction of the whole Web page by a crawler becomes smaller and smaller as the Web continues to grow. More and more dangling nodes appear because of the difficulty of sampling the entire Web. In [19], the authors reported that they have 51 million URLs not downloaded yet when they have 24 million pages downloaded. In [7], dynamic pages are estimated to be 100 times more than static pages, and in [4], the authors point out in their experiment that the number of uncrawled pages still far exceeds the number of crawled pages and that there are an essentially infinite number of URLs which is estimated to be at least 64^{2000}. These experimental results and theoretical analysis mean that in reality, the huge number of unvisited pages tends to exceed the ability of a crawler.

On the other hand, some dangling pages are worthy of ranking because they contain important information. In such a situation, ranking those pages that only have been found may enrich the content of a search engine. As an example, a search engine may return the users the URLs of unvisited pages with high ranking scores. Moreover, including dangling nodes in the overall ranking may have significant effect not only on the rank value of non-dangling pages but also on the rank order. This will be shown in the Experiment section.

How to Classify Dangling Nodes

In the following, we follow the ideas in [4] in analyzing the reasons that cause the dangling nodes, and we classify dangling nodes into 3 classes according to these reasons.

Dangling nodes of class 1 ($DNC1$) are defined as nodes that have been found but have not been visited. One reason to produce such kind of dangling nodes is that the Web is so large that we cannot visit all the pages; another

reason is that new Web pages are always being created.

Dangling nodes of class 2 ($DNC2$) are defined as nodes that have been tried but not visited successfully. The reason to produce dangling nodes of class 2 is that some pages may exist before, but now are damaged or are in maintenance, or they are protected by a robot.txt, or they are wrongly created.

Dangling nodes of class 3 ($DNC3$) are defined as nodes that have been visited successfully but from which no out-link is found. Dangling nodes of class 3 exist because there are many files on the Web with no hyperlink structure.

How to Handle Dangling Nodes

We first partition all the nodes V of the graph G ($|V| = n$) into three subsets: D^0, D^1, and D^2, where C^0 ($|C^0| = m$) denotes the subset of all nodes that have been crawled successfully and have at least one out-link; D^1 ($|D^1| = m_1$) denotes the set of nodes of $DNC3$; D^2 ($|D^2| = n - m - m_1$) denotes the set of nodes of $DNC1$. Nodes of $DNC2$ are ignored here. The main idea of handling dangling nodes is to handle different nodes in different ways. In the following, we describe our method in detail.

1. We predict the real in-degree $d^-(v_i)$ by the number of found links $fd^-(v_i)$ from visited nodes to the node v_i. With the breadth-first crawling method, we assume that the real number of links from all nodes in V to the node v_i is proportional to the number of found links $fd^-(v_i)$ from visited nodes to the node v_i, and further we assume that

$$d^-(v_i) \approx \frac{n}{(m+m_1)} \cdot fd^-(v_i)(i = 1, 2, \ldots, n).$$

This assumption is based on the intuition that a crawler's ability of finding new links to a given node v_i depends on the density of these links. The density of these links to the node v_i is equal to $d^-(v_i)/n$. The crawler has found $fd^-(v_i)$ such kind of links when it has crawled m nodes, and we consider $\frac{fd^-(v_i)}{(m+m_1)}$ as an approximate estimate of the density of these links. Following this, the above approximate equality holds.

2. With the approximate in-degree $d^-(v_i)$, we can re-arrange the matrix. All the found links $fd^-(v_i)$ are from the nodes in D^0, and the remaining links $d^-(v_i) - fd^-(v_i)$ are from the nodes in D^2 (it is impossible that some of these links are from the nodes in D^1). Since we infer the number of the remaining links only out of $m+m_1$ visited nodes and the total number nodes is n, there is a risk of over-prediction. To prevent the over-prediction, we adopt a confidence index (or certainty) $(m+m_1)/n$ about this estimation, and so we expect $(d^-(v_i)-fd^-(v_i))(m+m_1)/n$ remaining links. Without any prior information about the distribution of these remaining links, we have to assume that they are distributed uniformly from the nodes in D^2 to the node v_i, i.e., these remaining links are shared by

all the nodes in D^2. So matrix A^T representing the random graph can be divided into six blocks shown below

$$A^T = \begin{pmatrix} C & X & M \\ & & \\ D & Y & N \end{pmatrix},$$

where $(C, D)^T$ is used to model the known link structure from D^0 to V. Let $C = (c_{ij}), D = (d_{ij})$, then

$$c_{ij}, d_{i,j} = \begin{cases} 1, & \text{there is a link from } j \text{ to } i, \\ & \\ 0, & \text{otherwise.} \end{cases}$$

In A^T, $(X, Y)^T$ will be defined later, $(M, N)^T$ is used to model the link structure from D^2 to V, and is defined as follows:

$$\begin{pmatrix} M \\ \\ N \end{pmatrix} = \begin{pmatrix} l_1 & 0 & 0 & 0 \\ 0 & l_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_n \end{pmatrix} \mathbf{1}_{n\times(n-m-m_1)},$$

where $l_i = \frac{(d^-(v_i)-fd^-(v_i))(m+m_1)}{n(n-m-m_1)}$, $(i = 1, 2, \ldots, n)$, $n - m - m_1$ means that the expected remaining in-links $(d^-(v_i) - fd^-(v_i))(m + m_1)/n$ are shared uniformly by all nodes in D^2.

3. When we want to model the users' teleportation, we assume that the users will jump to node v_i with a probability of g_i when they get bored in following the actual links. So the matrix modelling the teleportation is ge^T. We denote here $(g_1\, g_2\, \ldots\, g_n)^T$ by g.

4. When the user encounters a node of $DNC3$, there is no out-link that the user can follow. In this case, we assume that the same kind of teleportation as in step 3 will happen, and so the matrix $(X, Y)^T$ in step 2 is used to model the link structure from D^1 to V and it is assumed to be

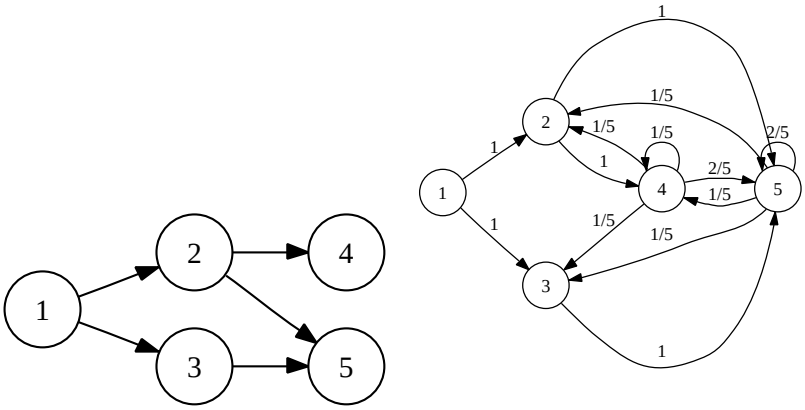
$$\begin{pmatrix} X \\ \\ Y \end{pmatrix} = \begin{pmatrix} g_1 & 0 & 0 & 0 \\ 0 & g_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g_n \end{pmatrix} \mathbf{1}_{n\times m_1}.$$

5. We further assume that α is the probability of following an actual out-link from a page, $1-\alpha$ is the probability of taking a "random jump" rather than following a link. Then the random matrix P is modelled as

$$P^T = (1 - \alpha)ge^T + \alpha A^T. \tag{5.3}$$

The matrix P corresponds to a random graph, which models the temporal Web–to predict a future Web graph by an early Web graph. This is called *Temporal Web Prediction Model*.

From the static graph in Figure 5.3(a), where node 4 and node 5 are assumed to be nodes of $DNC1$, a random graph in Figure 5.3(b) is generated by the above model ($\alpha = 1$).



(a) Original Static Graph (b) Random Graph produced

Figure 5.3. Illustration on the random graph

5.2.4 Random Graph Ranking

For *DiffusionRank*, we have extended it to random graphs. For the standard ranking algorithms, we also need to extend them to random graphs in order to handle the random graph outputs produced by the predictive strategy in three situations: (1) temporal links, (2) spatial links and (3) weighted links.

PageRank on a Random Graph

We extend the *PageRank* from the setting of a static graph to the setting of a random graph. Similar to *PageRank*, the page rank vector x on a random graph can be defined recursively in terms of random graphs:

$$x_i = \sum_j q_{ij} x_j,$$

where $q_{ij} = p_{ji} / \sum_k p_{jk}$. Or in matrix form, $x = Qx$, where $Q = (q_{ij})$. In a static graph, if there is a link from v_j to v_i, then the probability of a random surfer will follow the link is $1/d_j$, where d_j is the out-degree of v_j. In a random graph, since the sum $\sum_k p_{jk}$ is the expected out-degree of v_j and the link from v_j to v_i exists with a probability of p_{ji}, the expected probability of a random surfer will follow the link (v_j, v_i) is $p_{ji} / \sum_k p_{jk}$. Consequently, the above equation is established.

Common Neighbor on a Random Graph

We extend the *Common Neighbor* approach from the setting of a static graph to the setting of a random graph.

First, the random neighbor set $RI(v_i)$ of v_i is defined as

$$RI(v_i) = \{(v_k, p_{ki}) | v_k \in V\},$$

where p_{ki} is the probability of v_k as a neighbor of v_i. In the setting of a random graph, each node v_k is linked to node v_i with a probability p_{ki}, so v_k is the neighbor of v_i with a probability p_{ki}. This extends the definition of the set of neighbors of node v_i in the setting of a static graph.

Second, the set of the common random neighbors of v_i and v_j is defined as

$$RI(v_i) \cap RI(v_i) = \{(v_k, p_{ki}p_{kj}) | v_k \in V\}.$$

The sets of random neighbors of v_i and v_j are $RI(v_i)$ and $RI(v_j)$ respectively. v_k is the neighbor of v_i with a probability p_{ki}, and v_k is the neighbor of v_j with a probability p_{kj}, then we can say v_k is the common neighbor of v_i and v_j with a probability $p_{ki}p_{kj}$ since the random edges are drawn independently. This extends the meaning of the common neighbor.

Third, the expected number of nodes in $RI(v_i) \cap RI(v_i)$ is considered as the similarity measure $s(i, j)$, and is defined as

$$s(i, j) = \sum_k p_{ki}p_{kj}.$$

This extends the definition of number of common neighbors of v_i and v_j in the setting of a static graph.

Jaccard's Coefficient on a Random Graph

The *Jaccard's Coefficient* in the setting of a random graph is defined as

$$\begin{aligned} s(i, j) &= |RI(v_i) \cap RI(v_j)| / |RI(v_i) \cup RI(v_j)| \\ &= \textstyle\sum_k p_{ki}p_{kj} / \sum_k (p_{ki} + p_{kj} - p_{ki}p_{kj}), \end{aligned}$$

where $RI(v_i) \cup RI(v_j) = \{(v_k, p_{ki} + p_{kj} - p_{ki}p_{kj}) | v_k \in V\}$. The expected number elements in $RI(v_i) \cup RI(v_j)$ is equal to $\sum_k (p_{ki} + p_{kj} - p_{ki}p_{kj})$. Since v_k is not the neighbor of v_i with a probability $1 - p_{ki}$, and is not the neighbor of v_j with a probability $1 - p_{kj}$, we assume that v_k is not the neighbor of either v_i or v_j with a probability $(1 - p_{kj})(1 - p_{kj})$, and we have that v_k is the neighbor of either v_i or v_j with a probability $1 - (1 - p_{kj})(1 - p_{kj}) = p_{ki} + p_{kj} - p_{ki}p_{kj}$. Therefore, $RI(v_i) \cup RI(v_j) = \{(v_k, p_{ki} + p_{kj} - p_{ki}p_{kj}) | v_k \in V\}$. The expected number elements is thus equal to $\sum_k (p_{ki} + p_{kj} - p_{ki}p_{kj})$.

SimRank on a Random Graph

In the setting of a random graph, the *SimRank* score $s(i,j)$ of two pages v_i and v_j can be naturally redefined as the fixed point of the following recursive definition,

$$s(i,j) = \begin{cases} 1, & i = j, \\ \\ \frac{C}{|RI(v_i)||RI(v_j)|} \sum_{u,v} p_{ui} p_{vj} s(u,v), & others, \end{cases}$$

for some constant decay factor $C \in (0,1)$, where $|RI(v_i)| = \sum_k p_{ki}, |RI(v_j)| = \sum_k p_{kj}$.

Note that one can easily conclude that when the random graph becomes a static graph, the algorithms described in the above subsections degrade into the original algorithms. This means ranking algorithms on a random graph generalize the original ones.

Chapter 6

Experiments

6.1 Heat Diffusion Classifiers

The Parzen Window Approach (PWA), KNN, NHDC and PHDC are applied to six datasets from the UCI Repository. Table 6.1 describes the datasets we use. The first column refers to the names of the datasets, the second column refers to the number of cases in each dataset, the third column refers to the number of classes, and the fourth column is the number is the number of attributes. In the dataset Credit-g, we only consider the seven continuous attributes while the thirteen discrete attributes are ignored.

Table 6.1. Description of the Datasets

dataset	**Cases**	**Classes**	**Attributes**
Credit-g	1000	2	7
Diabetes	768	2	8
Glass	214	6	9
Iris	150	3	4
Sonar	208	2	60
Vehicle	846	4	18

In order to make each attribute in the same scale, we preprocess the datasets by transforming the domain of each attribute to the interval [0,1]. Specifically, for each attribute i, we transform the value x for attribute i by $(x - \min(i))/(\max(i) - \min(i))$, where $\min(i)$ and $\max(i)$ are the minimum and maximum value of attribute i, respectively.

The parameter setting is shown in Table 6.2. The figures shown in Table 6.3 are the mean error rates of ten-fold

cross-validations, and the last row in Table 6.3 shows the average results.

The experimental results show that NHDC uniformly outperforms PWA and KNN in accuracy, indicating the superiority of our approach. Furthermore, PHDC improves over NHDC.

Table 6.2. parameters setting of PWA KNN NHDC and PHDC

dataset	**PWA**	**KNN**	**NHDC**		**PHDC**		
	$1/\beta$	K	K	$1/\beta$	K	$1/\beta$	γ
Credit-g	50	31	13	0	11	0	0.02
Diabetes	300	34	33	50	34	150	0.05
Glass	7500	3	40	1750	38	1500	0.27
Iris	350	7	15	0	13	50	0.47
Sonar	1150	3	24	1650	24	1200	0.41
Vehicle	650	10	8	350	10	600	0.11

Table 6.3. Mean error rates of PWA KNN NHDC and PHDC

dataset	**PWA**(%)	**KNN**(%)	**NHDC**(%)	**PHDC** (%)
Credit-g	27.65	24.41	**23.90**	23.94
Diabetes	25.04	24.22	**23.70**	23.78
Glass	28.44	29.36	27.01	**26.88**
Iris	2.93	2.64	2.64	**2.21**
Sonar	11.72	17.14	11.25	**10.93**
Vehicle	27.55	28.59	27.10	**27.07**
Average	20.56	21.06	19.26	**19.14**

6.2 Predictive Random Graph Ranking

The temporal dimension of the *Predictive Random Graph Ranking* framework can actually be designed to be tested in experiments, although it is difficult to measure whether a link analysis algorithm is better than another because of the different intuitions for different ranking algorithms. For this, we design a comparison method by calculating the ranking difference and order difference between the early results (less accurate) and the final results (relatively accurate, and considered as a ground truth). For more details, see Section 6.2.2.

t	1	2	3	4	5	6
V[t]	1000	1100	1200	1300	1400	1500
T[t]	1764	1778	1837	1920	1927	1936
t	7	8	9	10	11	
V[t]	1600	1700	1800	1900	2000	
T[t]	1952	1954	1964	1994	2000	

Table 6.4. Description of the Synthetic Graph Series

6.2.1 Data

Our input data consists of a synthetic data set and a real-world data set. A detailed description follows.

Synthetic Web Graph

The degree sequences of the World Wide Web are shown to be well approximated by a power law distribution [11, 14, 13]. That is, the probability that a Web page has *k* outgoing (incoming) links follows a power law over many orders of magnitude $P_{out}(k) \sim k^{-\gamma_{out}}$ and $P_{in}(k) \sim k^{-\gamma_{in}}$.

The power law distribution of the degree sequence appears to be a very robust property of the Web despite its dynamic nature, therefore, we can generate synthetic Web-like random graphs to test the performance of our algorithms.

Several approaches to modelling power law graphs [11, 13] have been proposed. In our numerical experiment, we use the (α, β) model [13] to generate random graphs. By setting $\alpha = 0.52$ and $\beta = 0.58$, the model generates a random power law graph with $\gamma_{out} = 2.1$ and $\gamma_{in} = 2.38$, both of these values match the Web.

By simulating the procedure of crawling, we can obtain a series of growing incomplete graphs containing pages of $DNC1$. The number $V[t]$ of pages visited and the total number $T[t]$ of pages found at time t are shown in Table 6.4.

Real Web Graph

The data of a real Web graph were obtained from the domain *cuhk.edu.hk*. The graph series are snapshot during process of crawling pages restricted within this domain. The number $V[t]$ of pages visited and the total number $T[t]$ of pages found at time t are shown in Table 6.5.

t	1	2	3	4	5	6
V[t]	7712	78662	109383	160019	252522	301707
T[t]	18542	120970	157196	234701	355720	404728
t	7	8	9	10	11	
V[t]	373579	411724	444974	471684	502610	
T[t]	476961	515534	549162	576139	607170	

Table 6.5. Description of Real Data Sets Within Domain cuhk.edu.hk

6.2.2 Methodology

The algorithms we run include *PageRank* and *DiffusionRank*. For each algorithm A, we have two versions denoted by A and $PreA$. A is the original version without using the *Temporal Web Prediction Model* , and $PreA$ is the version using the *Temporal Web Prediction Model* . Both $PreA$ and A are run on two data series– the synthetic data series and the real data series. Each data series contains 11 data sets, which are obtained by taking snapshots during the process of a crawler or a simulated crawler. Finally, for each data series and for each algorithm A, we obtained 22 ranking results, namely,

$$\begin{array}{llll} A_1, & A_2, & \ldots, & A_{11}, \\ PreA_1, & PreA_2, & \ldots, & PreA_{11}. \end{array}$$

The results on the first 10 data is not accurate because these data are incomplete, and the Web is dynamically changing. The result A_{11} on the synthetic data should be the same as $PreA_{11}$ because *Temporal Web Prediction Model* will not have effect on complete information, but the result A_{11} on the real data is not the same as $PreA_{11}$ because of the existence of dangling nodes of $DNC1$ in time 11.

If the difference between the results on time t and the results on time 11 is smaller, we think it is more accurate. We calculate the value difference and order difference described below.

Value Difference. The value difference between A_t ($PreA_t$) and A_{11} is measured as

$$||A_t/Max_t - Cut(t, A_{11})/CutMax_t||_2$$

($||PreA_t/Max_t - Cut(t, A_{11})/CutMax_t||_2$). Where $cut(t, A_{11})$ is the results cut from A_{11} such that it has the same dimension as A_t, and $CutMax_t$ (Max_t) means the maximal value among results in $cut(t, A_{11})$ (A_t).

Order Difference. The order difference between A_t ($PreA_t$) and A_{11} is measured as the significant order difference between A_t and $Cut(t, A_{11})$ ($PreA_t$ and $Cut(t, A_{11})$). The significant order difference between two

similarity matrices M and N is calculated by the sum of the significant order difference for each row of M and N, and for each row $M(i), N(i)$ of M and N, the pair $(M(i,j), M(i,k))$ and $(N(i,j), N(i,k))$ is considered as a significant order difference if both $M(i,j) > M(i,k) + 0.005Max_M$ and $N(i,k) > N(i,j) + 0.005Max_N$, or both $M(i,k) > M(i,j) + 0.005Max_M$ and $N(i,j) > N(i,k) + 0.005Max_N$, where Max_M (Max_N) is the maximum value of M (N).

6.2.3 Set Up

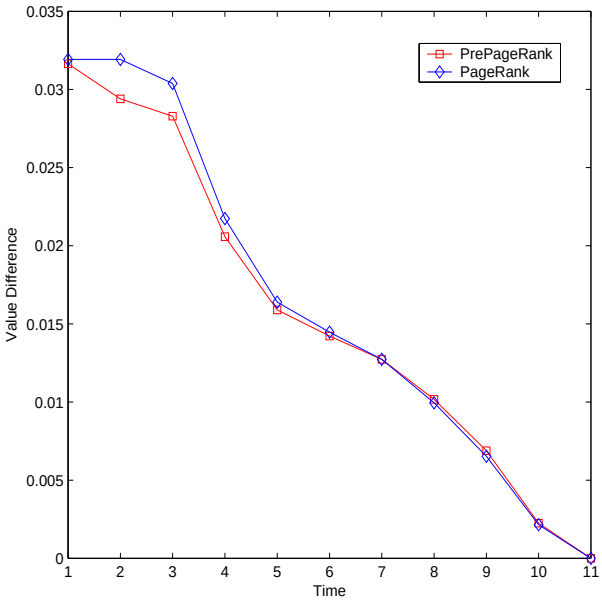
The experiments are conducted on the workstation whose hardware model is Nix Dual Intel Xeon 2.2GHz, whose RAM is 1GB, and whose OS is Linux Kernel 2.4.18-27smp (RedHat7.3).

We set $\alpha = 0.85$ and set g to be the uniform distribution in both *PageRank* and *PrePageRank*. Note that we use the modified *PageRank* algorithm [10], in which dangling nodes of $DNC1$ are considered to have random links uniformly to each node. For *DiffusionRank* and *PreDiffusionRank* , we use the *Discrete Diffuse Kernel* for computing, and we set $\sigma = 1$, $N = 20$ and β to be the inverse of the maximal out-degree in both. Note that *DiffusionRank* uses the *Discrete Diffuse Kernel* on a static graph and *PreDiffusionRank* uses *Discrete Diffuse Kernel* on a random graph.

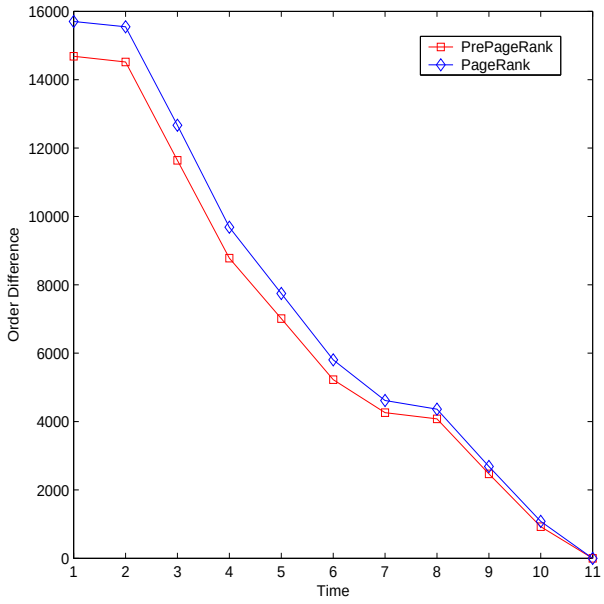
6.2.4 Experimental Results

Figure 6.1 demonstrate the *PageRank* results on the synthetic data and the real data. On the synthetic data, in 60% early stages, *PreRageRank* is closer to the final result in value difference; in 100% early stages, *PrePageRank* is closer to the final result in significant order difference. Since the graph in time 11 is complete, there is no difference between the *PrePageRank* and *PageRank*, and the curves meet at time 11. On the real data, since the data at time 11 contains unvisited pages, *PrePageRank* and *PageRank* have a difference on this data, the employment of *PageRank* results on this data as a reference will cause a bias against *PrePageRank*. Even so, in 60% early stages, *PreRageRank* is closer to the final *PageRank* result in value difference; in 70% early stages, *PrePageRank* is closer to the final *PageRank* result in significant order difference.

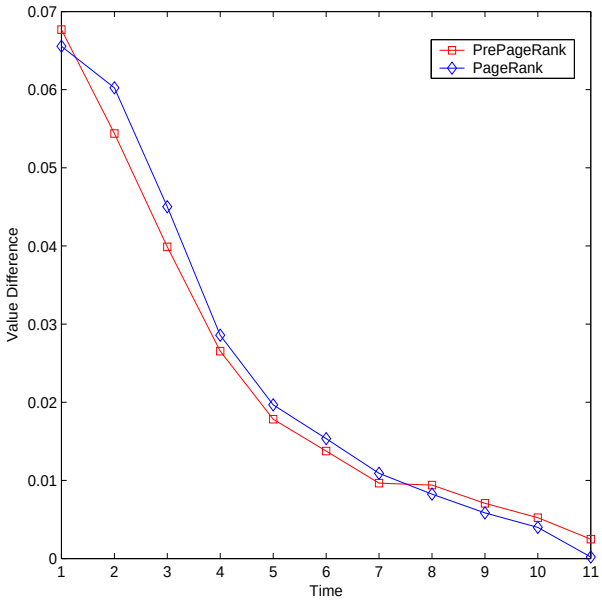
Figure 6.2 demonstrate the *DiffusionRank* results. On the synthetic data, in 100% early stages, *PreDiffusion-Rank* is closer to the final result in value difference, and in 100% early stages, *PreDiffusionRank* is closer to the final result in significant order difference. On the real data, since the data at time 11 contains unvisited pages, *PreDiffusionRank* and *DiffusionRank* have a difference on this data, the employment of *DiffusionRank* results on this data as a reference will cause a bias against *PreDiffusionRank*. Even so, in 70% early stages, *PreDiffusion-Rank* is closer to the final *PageDiffusionRank* result in value difference; in 70% early stages, *PreDiffusionRank* is



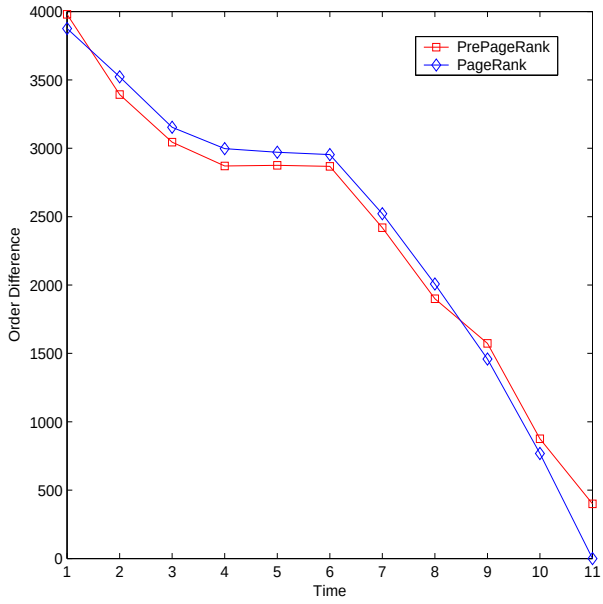
(a)-VD in synthetic data



(b)-OD in synthetic data



(c)-VD in real data



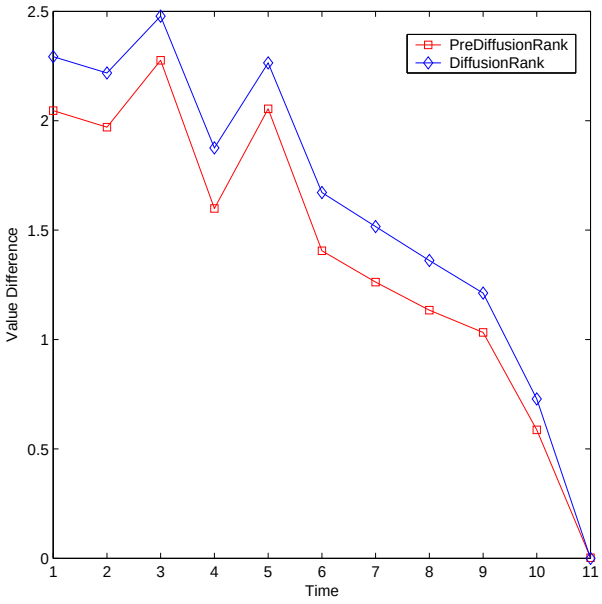
(b)-OD in real data

Figure 6.1. PageRank Comparison Results

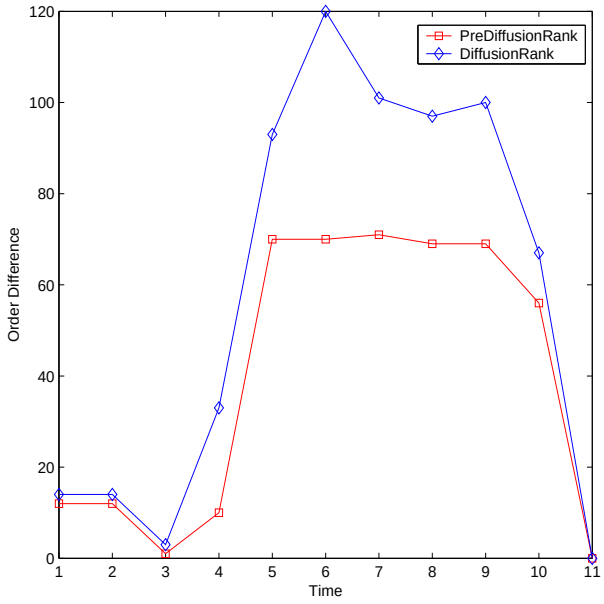
closer to the final *DiffusionRank* result in significant order difference.

6.2.5 Discussion

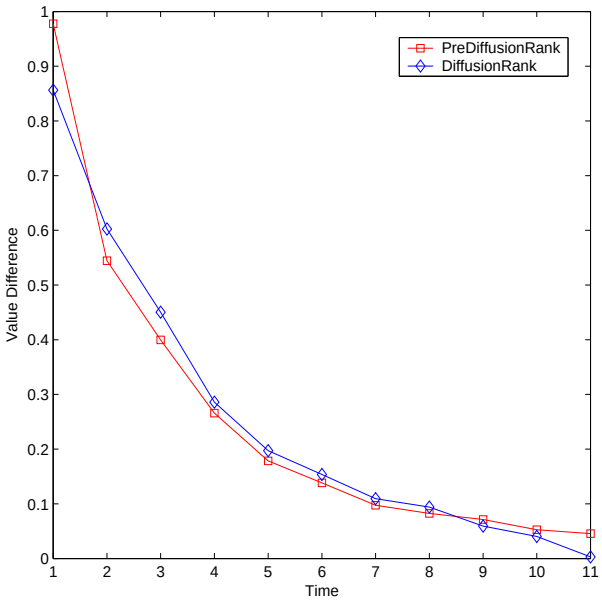
For *Common Neighbor*, *Jaccard's coefficient*, and *SimRank*, similar experiments are conducted. On *Common Neighbor*, slight improvement is achieved, but on *Jaccard's coefficient* and *SimRank* we do not obtained expected results for the *Temporal Web Prediction Model*. These abnormal results may be caused by the ignorance of the power law distribution. Before *Temporal Web Prediction Model*, the data satisfies the power law distribution, but after the model, the in-degrees of all nodes are increased with the same proportions. This in fact breaks the power law distribution, for example, nodes whose in-degree is 1 do not exist after prediction while nodes whose in-degree is 1 should have the highest density according to the power law distribution. *Jaccard's coefficient* and *SimRank* seem to be sensitive to the distribution of in-degrees and out-degrees. It is interesting and challenging to preserve the power law distribution in the *Temporal Web Prediction Model* so that better accuracy can be achieved on all these algorithms.



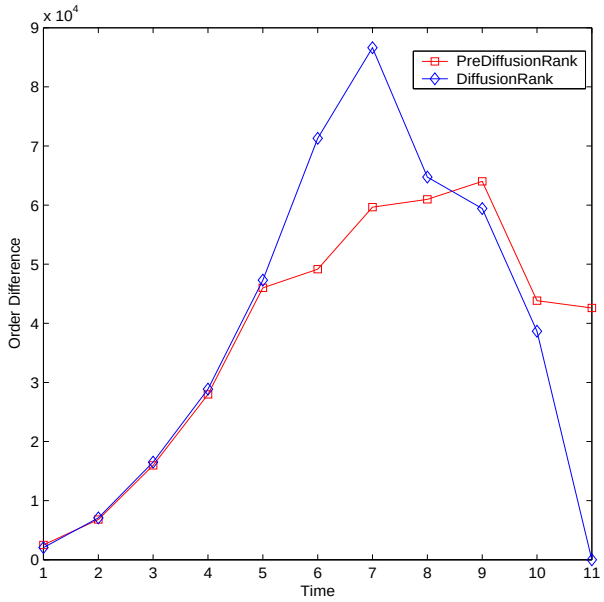
(a)-VD in synthetic data



(b)-OD in synthetic data



(c)-VD in real data



(b)-OD in real data

Figure 6.2. DiffusionRank Comparison Results on small synthetic data

Chapter 7

Conclusion and Future Work

We have presented two classifiers NHDC and PHDC by imitating the way that heat flows in a medium with a geometric structure. By approximating the manifold by the K nearest neighbors graph, we can avoid the difficulty of finding the explicit expression for the unknown geometry in most cases. By establishing the heat diffusion equation on the graph, we avoid the difficulty of finding a closed form heat kernel for some complicated geometries. Moreover, our solution to heat equation has the property of heat preserving, but our heat kernel is not symmetric and positive definite. While NHDC is a generalization of both Parzen Window Approach (when the window function is a multivariate normal kernel) and KNN, PHDC can approximate NHDC if parameter γ is small. Both NHDC and PHDC are proven to be efficient in our experiments.

We have shown that *DiffusionRank* is another candidate of ranking algorithms, and we have shown that the *Temporal Web Prediction Model* is effective in *PageRank* and *DiffusionRank*. Because our model mines more information about the Web structure, the results of Predictive strategy on these two algorithms are more accurate than those without it, even our model breaks the power law distribution. We conclude that the random graph input indeed extends the scope of some original ranking techniques, and significantly improve some of them.

In our experiments, we only test the *Predictive Random Graph Ranking* framework in the viewpoint of dynamic Web. Besides the challenging work to consider the power law distribution in this viewpoint, it deserves further investigation in other two viewpoints of partial observers and weighted links. Such future work involves investigating page-makers' preference on link orders and substantial users-based research.

Acknowledgments

I thank Prof. Irwin King and Prof. Michael R. Lyu for their kind guidance. I thank Mr. Patrick Lau, Mr. Zhenjiang Lin and Mr. Zenglin Xu for their help. I thank Prof. Miroslaw Malek for his useful comments. The UCI

Data Repository owes its existence to David Aha and Patrick Murphy.

Appendix

Some general terms are shown below.

v_i:	node with an index i
V:	set of nodes
(v_i, v_j):	edge from v_i to v_j
E:	set of edges
$G = (V, E)$:	a graph
$RG = (V, P)$:	a random graph
x_i:	rank value for node v_i
x:	rank vector consisting of x_i
$s(i, j)$:	similarity score for v_i and v_j
$I(v_i)$:	set of nodes that have links to v_i
$RI(v_i)$:	random neighbor set
$DNC1$:	dangling node of class 1, which is not visited but has been found by a crawler
$DNC2$:	dangling node of class 2, which has been tried but not visited successfully
$DNC3$:	dangling node of class 3, which has been visited successfully and from which no out-link is found
$d^+(v_i)$:	out-degree of node v_i
$d^-(v_i)$:	in-degree of node v_i
$f_i(t)$:	heat at v_i at time t
$(I + \sigma \Delta t H)^N$:	discrete diffusion kernel on a static graph
$e^{\sigma H}$:	continuous diffusion kernel on a static graph
$(I + \sigma \Delta t R)^N$:	discrete diffusion kernel on a random graph
$e^{\sigma R}$:	continuous diffusion kernel on a random graph

Bibliography

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] B. Bollobás. *Random Graphs*. Academic Press Inc. (London), 1985.

[4] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceeding of the 13th World Wide Web Conference*, pages 309–318, 2004.

[5] A. Ellis and T. Hagino, editors. *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14*. ACM, 2005.

[6] D. Fogaras and B. Rácz. Scaling link-based similarity search. In Ellis and Hagino [5], pages 641–650.

[7] S. Handschuh, S. Staab, and R. Volz. On deep annotation. In *Proceeding of the 12th World Wide Web Conference*, pages 431–438, 2003.

[8] H. Ino, M. Kudo, and A. Nakamura. Partitioning of web graphs by community topology. In Ellis and Hagino [5], pages 661–669.

[9] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. *Proc. of SIGKDD*, 2002.

[10] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub. Exploiting the block structure of the web for computing pagerank. Technical report, Stanford University, 2003.

[11] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, and A. S. Tomkins. The Web as a graph: Measurements, models and methods. *Lecture Notes in Computer Science*, 1627:1–18, 1999.

[12] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2002.

[13] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the web. In *The VLDB Journal*, pages 639–650, 1999.

[14] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks (Amsterdam, Netherlands)*, 31(11–16):1481–1493, 1999.

[15] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, 2005.

[16] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Twelfth International Conference on Information and Knowledge Management*, pages 556–559. ACM, November 2003.

[17] C. R. MacCluer. The many proofs and applications of perron's theorem. *SIAM Review*, 42(3):487–498, 2000.

[18] M. E. J. Newman. Scientific collaboration networks. I. Network construction and fundamental results. *Physical Review E*, 64(016131):1–8, 2001.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report Paper SIDL-WP-1999-0120 (version of 11/11/1999), Stanford Digital Library Technologies Project, 1999.

[20] S. Rosenberg. *The Laplacian on a Riemmannian Manifold*. Cambridge University Press, 1997.

[21] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(22):2323–2326, 2000.

[22] L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensinal manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[23] J. B. Tenenbaum, V. d. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22):2319–2323, 2000.

[24] H. Yang, I. King, and M. R. Lyu. Predictive ranking: a novel page ranking approach by estimating the web structure. In A. Ellis and T. Hagino, editors, *WWW (Special interest tracks and posters)*, pages 944–945. ACM, 2005.