Efficient Algorithm for Handling Dangling Pages Using Hypothetical Nodes

Ashutosh Kumar Singh  
Department of Electrical and Computer Engineering  
Curtin University of Technology  
Miri, Malaysia  
ashutosh.s@curtin.edu.my

Ravi Kumar P  
Department of Electrical and Computer Engineering  
Curtin University of Technology  
Miri, Malaysia  
ravi2266@gmail.com

Alex Goh Kwang Leng  
Department of Electrical and Computer Engineering  
Curtin University of Technology  
Miri, Malaysia  
avexoh.kwangleng@gmail.com

Abstract—Web is growing in an incredible fashion as the only source of knowledge. Search engines are the main tools used in knowledge extraction or information retrieval from the Web. Google is one of the top search engines used by the knowledge seekers. PageRank algorithm is used in the famous Google search engine and is one of the best ranking algorithms that is currently in use. There are some problems explored by the researchers in the PageRank Algorithm. One such problem is the Dangling Pages or sinks. This paper focus on the dangling pages and the problems associated with dangling pages and how it affects the PageRank results. There are a couple of solutions discussed for the dangling pages and we implemented one method.

Keywords—PageRank, Dangling Pages.

I. INTRODUCTION

The PageRank algorithm [1] of Page et al. has taken a new step in ranking of the results produced by search engines. The PageRank algorithm is a link-based ranking method and it is used in the famous Google search engine. Even though there are lot of good things about PageRank algorithm, there are also some problems associated with the PageRank. One such problem is the “dangling page” i.e. web pages not having any outgoing links or pages without any hyperlinks in the web. It can also be called as hanging pages. Such hanging pages can act as sinks or black holes for PageRank, a key factor in the Google search algorithm. For example if we consider the following snap shot (Figure 1).

![Figure 1 A snapshot of a web](image)

This dangling page problem is discussed by many researchers Page et al. [1], N. Eiron et al. [2], Amy N. Langville et al. [3], Xuanhui Wang et al. [4].

This paper is organized as follows. Next Section shows PageRank model, the PageRank algorithm and the calculation of the PageRank. Section III provides details on “dangling page” and the previous work on the dangling page. Section IV explains our work on the dangling page and the experimental results. Paper is concluded in Section V.

II. PREVIOUS WORK

A. Overview

PageRank algorithm [1] is developed by Brin and Page of Stanford University during their Ph D using Citation Analysis [6, 7]. The basic PageRank model treat the whole Web as a directed graph \( G(V,E) \), with a vertex set of \( V \) of \( N \) pages and a directed edge set \( E \). PageRank importance or rank is determined by the “votes” in the form of in-links from other pages of the Web. The links (votes) from important sites carries more weight than links (votes) from less important sites. Assume any arbitrary page \( A \) has pages \( T_1 \) to \( T_n \) pointing to it (incoming link). PageRank can be calculated by the following equation (1).

\[
PR(A) = (1 - d) + d(PR(T_1) / C(T_1) + \ldots + PR(T_n) / C(T_n))
\]  

(1)

The parameter \( d \) is a damping factor, usually sets it to 0.85 (to stop the other pages having too much influence, this total vote is “damped down” by multiplying it by 0.85). \( C(A) \) is defined as the number of links going out of page \( A \). The PageRank is treats the web surfing as a Markov chain. An arbitrary collection of web pages indexed as \( \mathbb{S} = \{1,2,\ldots,N\} \) and the personalization vector \( u \in \mathbb{R}^{N \times 1} \) which signifies a generic surfer’s preference for each page in \( \mathbb{S} \). Let this generic surfer be currently at page \( i \in \mathbb{S} \). In the next step the surfer may move to some to some \( j \in \mathbb{S} \) according to the probability:

\[
Q(i, j) = \begin{cases} 
\frac{G(i, j)}{\sum_{l=1}^{N} G(i, l)} & \text{if } G(i, j) = 1 \text{ for some } m \in \mathbb{S}; \\
ą(j) & \text{otherwise}
\end{cases}
\]
where \( G(i, j) = \begin{cases} 1 \\ 0 \end{cases} \)

The above definition says that the \( i \)-th page has outlinks, then the surfer can move to one of the links with an equal probability; if no outlinks \( i \) exist, the surfer can move to any page in \( \mathbb{S} \) at probabilities according to preference. A page has no outlink can be called as *dangling page*. Let us consider a sample small dating will converge to every page in the Web is not.

The Markov model of the Web is used to make the transition probability matrix, which is built from the hyperlink structure of the Web and it is stochastic and primitive. It treats the hyperlink structure of the Web as a directed graph (digraph). The nodes of this digraph represents web page and the directed arcs represents hyperlinks. Figure 3 shows a sample Web Page with 6 pages.

![A simple Node-Link graph for small 4 node web](image)

Figure 2  A simple Node-Link graph for small 4 node web

The *PageRanks* form a probability distribution over the Web pages, so the sum of all Web pages’ *PageRank* will be one. *PageRank* can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the Web.

Assume \( M \) is the row normalized matrix of \( A \), the updating of PageRank scores can be given as

\[
 r = M^T r
 \]  

When there is at least one nonzero entry in each row, mathematically the recursive updating will converge to \( M \)'s principal eigenvector. However the convergence is guaranteed only if \( M \) is irreducible and aperiodic [8]. A matrix is irreducible if its graph shows that every node is reachable from every other node. A nonnegative and irreducible matrix is primitive if it has only one eigenvalue on its spectral circle. An irreducible Markov chain with a primitive transition matrix is called an aperiodic chain. In Web applications aperiodic is guaranteed and the irreducible is not it is because every page in the Web is not connected to every other page. This will result in “rank sink” problem. To avoid this problem PageRank uses a uniform matrix \( U \ (U_{ij} = 1/N) \) and interpolates it with the original matrix \( M \) with a damping factor \( 1 - \alpha \) as follows:

\[
 \tilde{M} = (1 - \alpha) M + \alpha U
 \]  

Where \( \alpha \ (0 \leq \alpha \leq 1) \) is the random jumping probability. The revised PageRank score can be calculated using the following formula.

\[
 r = M^T r = \left(1 - \alpha \right) M^T r + \alpha \frac{e_n}{N} \]  

Where \( e_n = (1, ..., 1)^T \) is a column vector consisting of \( N \) elements of 1 and \( \alpha \) is set to 0.15 [1]. This reason behind the interpolation can be explained by a random surfing model. A surfer follows the out-links of a page with probability \( 1 - \alpha \) and uniformly jumps to random pages with probability \( \alpha \). A page’s PageRank score can be interpreted as the average probability that a surfer would visit this page after surfing the whole web for a period of time.

B. Markov model of the Web

The Markov model of the Web is used to make the transition probability matrix, which is built from the hyperlink structure of the Web and it is stochastic and primitive. It treats the hyperlink structure of the Web as a directed graph (digraph). The nodes of this digraph represents web page and the directed arcs represents hyperlinks. Figure 3 shows a sample Web Page with 6 pages.

![A Directed Graph showing 6 pages of web](image)

Figure 3  A Directed Graph showing 6 pages of web

The Markov model represents the above graph in Figure 3 with a square matrix \( P \) whose element \( p_{ij} \) is the probability of moving from state \( i \) (page \( i \)) to state \( j \) (page \( j \)) in one step or click. For the above graph in Figure 1, starting from any node (page) it is equally likely to follow any of the outgoing links to arrive at another node. So,

\[
 P = \begin{pmatrix}
 X1 & X2 & X3 & X4 & X5 & X6 \\
 X1 & 0 & 1/4 & 0 & 1/4 & 1/4 & 1/4 \\
 X2 & 1/2 & 0 & 1/2 & 0 & 0 & 0 \\
 X3 & 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\
 X4 & 0 & 0 & 0 & 0 & 0 & 0 \\
 X5 & 0 & 0 & 0 & 0 & 0 & 0 \\
 X6 & 1/3 & 0 & 1/3 & 0 & 1/3 & 0
 \end{pmatrix}
 \]

Any suitable probability may be used across the rows. The random surfer accessing page X1 may jump to page X2, X4, X5 or X6 and the random jumping probability to page X2 is only
25%. The fourth and fifth rows in the matrix are entirely zero because there are no outlinks from pages X4 and X5 and consequently \( P \) is not a stochastic matrix. One remedy is to replace all zero rows, \( 0^T \), with \( \frac{1}{n} e^T \), where \( e^T \) is the row vector of all ones and \( n \) is the order of the matrix. The revised transition probability matrix called \( \bar{P} \) is

\[
\bar{P} = \begin{pmatrix}
0 & 1/4 & 1/4 & 1/4 & 1/4 \\
1/2 & 0 & 1/2 & 0 & 0 \\
0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\
1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\
1/3 & 0 & 1/3 & 0 & 1/3 & 0
\end{pmatrix}
\]

Now this matrix \( \bar{P} \) is stochastic but it is not irreducible. To make it irreducible apply the formulae

\[
\bar{P} = \alpha \bar{P} + (1 - \alpha) \frac{1}{n} e^T
\]

### III. Dangling Pages

#### A. Dangling Pages

Dangling pages are those Web pages they don’t have any outgoing links or if the outgoing links are unknown but have incoming links i.e. the pages are being referenced by other pages and they are not referring to other pages. It is also referred as zero-out-link [4] problem, web frontier [2]. When the original PageRank algorithm was developed, the Web is mostly a static one using human edited documents residing in the file systems. So it is easy to crawl and index the entire Web. Recently Web is evolving into a dynamic one driven by databases. The crawlers can not crawl the entire Web particularly the frontier of the Web.

Dangling pages may appear in a Web for a variety of reasons. A page might have protected by robots.txt and crawlers can not visit those pages. Those pages may have high quality information and it might be useful for researchers. It is still possible to index those pages using anchor text even though crawlers can not crawl to those pages. Another reason for dangling pages are those pages they do not have outlink like the PDF files, pictures etc and those content might have useful information. The third type dangling pages is the URLs with meta tag indicating that links should not be followed from the page or it may require authentication. The fourth reason for dangling pages is those pages return a 500 class response at crawl time due to configuration problems and network problems. According to N. Eiron et al. [2], the number of dangling pages in the Web is keep growing. So dangling pages can not be omitted, they may contain quality information and one of the objective of knowledge extraction is not to miss out any information from any sources.

#### B. Problems associated with Dangling Pages

There are a number of problems associated with dangling pages. Some problems are direct like Link rot. Link rot problem is like the links are worked at one time are broken by the removal of content on the Web or the URL is being changed. This will return with HTTP code 403 or 404. The pages that return this 403 or 404 HTTP code are called as penalty pages. This penalty pages may have the effect [2] on the ranks of nearby pages. There are several algorithms “push-back”, “self-loop”, “jump-weighting” and BHITS” proposed by N. Eiron et al. [2] based on the basic PageRank algorithm to adjust the ranks of pages with links to penalty pages. There is another indirect problem associated with dangling pages is “zero-one-gap” problem proposed by Xuanhui Wang et al. [4]. The zero-one-gap problem refers to the number of out-links a page has. According to the authors there is a big gap between pages having 0 and 1 out link. This is because of the probability of jumping to random pages is 1 in a zero-out-link page and this drops to \( \alpha \) for a page with a single out-link. This zero-one-gap problem allows the spammers to easily manipulate PageRank results by setting up bogus link pages to solely increase the PageRank results. Xuanhui Wang et al. [4] proposed a DirichletRank algorithm based on the Bayesian estimation of transition probabilities. This DirichletRank not only solves the problem of zero-one-gap, but also provides a solution to solve the zero-out-link problem i.e. the dangling pages.

The presence of the dangling pages can cause philosophical, storage and computational problems for PageRank. The philosophical problems refer to the removal of the dangling pages during the computation of the PageRank and then add them back after the convergence of the PageRank (in the Brin and Page original PageRank algorithm). Some dangling nodes may get high PageRank and the removal of those nodes is not fair. For example, a very quality pdf file may have many inlinks from good sources and it should receive a high rank. It is not justice on that dangling page point of view. Incorporating the dangling pages [3] adds only a little computational effort but provides a fair computation of PageRank. The storage problem can be handled by constructing one vector \( \alpha \) implicitly, instead of the stochastic fix model proposed by Brin and Page. Element \( x_i = 1 \) if row of \( P \) corresponds to a dangling node, and 0, otherwise. The computational problem refers to the rate of convergence and the number of iterations.

#### C. Dealing with Dangling pages

Here we discussed some of the previous methods for handling dangling pages. In the original PageRank algorithm [1] proposed by Brin and Page, the dangling pages are removed from the graph and the PageRank is calculated for the non-dangling pages. After calculating the PageRank, the dangling pages can be added back in without affecting the results. The authors claim that few iterations are enough to remove most of the dangling pages. S.D. Kamvar et al. [9] suggested by removing the dangling nodes and then re-inserting them for the last few iterations. Completely removing all the dangling pages will alter the results on the non-dangling pages to some extent since the outdegrees from the pages are adjusted to reflect the lack of links to dangling pages. This approach is supported by S. Brin et al. [10] and T. Haveliwala [11]. N. Eiron et al. [2] keep all the dangling pages and calculate the ranking. S.D. Kamvar et al. [9] suggested that to jump to a randomly selected page with
probability 1 from every dangling page. For example the nodes V of the graph (n = |V|) can be partitioned into two subsets:

- S corresponds to a strongly connected subgraph (|S| = m).
- The remaining nodes in subset D have links from S but no outlinks.

A virtual node (n+1) is added to which the random jumps may be made. The new node set is denoted by V' = V U {n + 1}. New edges (i, n + 1) are added. In that i ∈ D and (n + 1, j) for j ∈ S to define an expanded edge set e'. PageRank of the nodes in V' can be computed via the principal eigenvector by partitioning the matrix and vector.

\[
\begin{bmatrix}
\alpha S & O & e/m \\
\alpha D & O & 0 \\
(1 - \alpha) e^T & e^T & 0
\end{bmatrix}
\begin{bmatrix}
p \\
q \\
r
\end{bmatrix}
= \begin{bmatrix}
p \\
q \\
r
\end{bmatrix}
\]

(5)

where, if d_j is the out degree of node j

\[
s_{ij} = \begin{cases} 
d_j^{-1} & \text{if } (i, j) \in e \text{ and } i, j \in S \\
0 & \text{otherwise}
\end{cases}
\]

\[
d_{ij} = \begin{cases} 
d_j^{-1} & \text{if } (i, j) \in e \text{ and } i \in S, j \in D \\
0 & \text{otherwise}
\end{cases}
\]

and p, q, r are of the row dimension of S and D and 1 and e is the vector of 1's of conforming dimension. The individual equations are:

\[
p = \alpha Sp + (r / m)e
\]

(6)

\[
q = \alpha Dp
\]

(7)

\[
r = (1 - \alpha) p + e^T q
\]

(8)

\[
= \left( (1 - \alpha) e^T + \alpha e^T D \right) p
\]

(9)

This structure can be used to compute p and q from a reduced eigen-system.

\[
\begin{bmatrix}
p \\
q \\
r
\end{bmatrix}
= \left( (1 - \alpha) e^T + \alpha e^T D \right)^e \begin{bmatrix}
p \\
q \\
r
\end{bmatrix}
\]

(10)

Since the reduced matrix is column stochastic and

\[
p = \alpha S p + \left( r / m \right)e
\]

(11)

\[
r = \left( (1 - \alpha) e^T + \alpha e^T D \right) p
\]

(12)

Now p and r can be solved in equation (10) by a standard iterative method like power iteration and then q can be calculated.

\[
q = \alpha D p
\]

(13)

A fast two-stage algorithm for computing PageRank and its extensions is proposed by Chris Pan-Chi Lee et al [12] is based on the Markov chain reduction. The PageRank vector is considered as the limiting distribution of a homogeneous discrete-time Markov chain that transforms one web page to another web page. The authors provided a fast algorithm for computing this vector. This algorithm uses the “lumpability” of Markov chain and constructed in two stages. In the first stage, the authors compute the limiting distribution of a chain where only the dangling pages are combined into one super node. In the second stage, the authors compute the limiting distribution of a chain where only the non-dangling pages combined. When this two limiting distributions are concatenated, the limiting distribution of the original chain, the PageRank vector is produced. According to the authors, this method can dramatically reduce the computing time and is conceptually elegant.

IV. EXPERIMENTAL RESULTS

We used the sample directed graph with 6 nodes shown in Fig. 1 as the input to our PageRank program, implemented in Java Program. It is a simple program but it is very effective in producing the PageRank results. Because of the limited resources we could not apply the PageRank program on a real Web, instead we just used the sample Graph shown on Fig. 1. The graph also includes two dangling pages X4 and X5.

Program Pseudo Code

======================================
Main Procedure
Initialize checkIteration is true
DO
  call PageRank to calculate the PageRank for every nodes
  save the PageRank for every nodes
  If the PageRanks of last Iteration has the same PageRanks with current Iteration
    checkIteration is false
  WHILE quits when checkIteration is false

Procedure PageRank
  Initialize result to 0.15 (1 - the damping factor)
  FOR every outgoing nodes of the current node
    call Calc
  Add up result with the results from Calc of all outgoing nodes

Procedure Calc
  Calculate the result by getting the PageRank of the current node divide by the numbers of outgoing links of the current node times 0.15 (1 - the damping factor)
V. CONCLUSION

This paper covers mathematical model of the PageRank algorithm. Markov model of the Web is explained and of Web mining. The importance of the Web structure mining in Information retrieval is explained. The main purpose of this paper is to explore the hyperlink structure and understand the Web graph in a simple way. This paper also focuses on the important algorithms used for hyperlink analysis, explore those algorithms and compare them.

REFERENCES


This paper is an accepted manuscript in 2010 6th International Conference on Digital Content, Multimedia Technology and its Applications (IDC).

The original published manuscript can be found in:

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5568585