Efficient Methodologies to Handle Hanging Pages Using Virtual Node

Ashutosh Kumar Singh, Ravi Kumar P, Alex Goh Kwang Leng
Department of Electrical and Computer Engineering
Curtin University, Sarawak Campus, Miri, Malaysia.
ashutoshs@curtin.edu.my, ravi2266@gmail.com, alex.goh@curtin.edu.my

In this paper we first explain the Knowledge Extraction (KE) process from World Wide Web (WWW) using Search engines. Then we explore the PageRank algorithm of Google Search engine (one of the famous link based search engine) with its hidden Markov analysis. In that we also explore one of the problems of Link based ranking algorithms called hanging pages or dangling pages (pages without any forward links). The presence of these pages affects the ranking of Web pages. Some of the hanging pages may contain important information which cannot be neglected by the search engines during ranking. We proposed methodologies to handle the hanging pages and compare the methodologies. We also introduce TrustRank algorithm (an algorithm to handle the spamming problems in link based search engines) and included it in our proposed methods so that our methods combat Web spam. We implemented PageRank algorithm and TrustRank algorithm and modified those algorithms to implement our proposed methodologies.

Keywords: Information Retrieval, Knowledge Extraction, Link Based Ranking Algorithms, Hanging Nodes, PageRank, TrustRank, Markov analysis

INTRODUCTION

Information Retrieval (IR) from the World Wide Web has changed the whole scenario of the traditional Knowledge Extraction process. Web users and the researchers started believing that the knowledge extracted from the Web is authentic and relevant. The first one is correct and the second one is
only partially correct because there are some information is not fully extracted by the search engines. Search engines are the main tools used in knowledge extraction or Information retrieval from the Web. There may be more relevant and important information for a given topic in the Web. But it is up to the ranking algorithms of the Search engines to decide which information to appear first. This makes the ranking algorithms in the search engine becomes more important. The PageRank algorithm (Page et al. 1998) has taken a new step in ranking of the results produced by search engines. Among several ranking algorithms, PageRank is the most efficient link-based ranking algorithm and it is used by the popular Google search engine. In PageRank, the ranking order is purely determined by the link structure of the Web and not by the content of the sites. PageRank algorithm is influenced by the famous citation analysis (Garfield 1972 and Pinski et al. 1976) used in the information science as a tool to identify core sets of articles, authors, or journals of particular field of study. Link analysis is based on the citation analysis.

A random surfer normally surf the Web going from one page to another page by randomly choosing a forward link. When a page does not have any forward link or when the surfer gets bored then he/she chooses a page by other then selecting the forward link. When a page is not having any forward or outgoing link then that page is called as hanging or dangling page. This hanging pages are one of the hidden problems of link based ranking algorithms because they don’t propagate the rank scores to other pages as it is important in link based ranking methods. Hanging pages in the Web can be occurred in four ways: (i) pages that are protected by robots.txt file, (ii) pages not having genuine forward link like the pdf files, images etc. (iii) pages that went into 5xx server error (Link Rot) and (iv) URLs with meta tag indicating that the links should not be followed. Hanging pages keep growing in the Web (Eiron et al. 2004) and they cannot be left out during the ranking process because they may contain quality information.

According to Langville et al. (2003), the theoretical random walk of the Web can be considered as Markov Chain or Markov process. The limiting probability that a dedicated random surfer visits any particular page is its PageRank. A page has higher PageRank if it has links to and from other pages with high rank. Technical reports on PageRank algorithm is shown by Page et al. (1999), Langville et al.
(2003) and Ridings et al. (2002), Page et al. (1998), Eiron et al. (2004), Langville et al. (2005), Bianchini et al. (2005), Wang et al. (2008) and Pan-Chi Lee et al. (2003) discussed about hanging pages. This paper is organised as follows. The next section brief about the previous work on hanging pages. After that our proposed methods are explained in detail with examples followed by the experimental results. The last section concludes this paper with future works.

**PREVIOUS WORK**

Here we discussed some of the previous methods for dealing with hanging pages. In the original PageRank algorithm proposed by Brin and Page (1998), the hanging pages are removed from the graph and the PageRank is calculated for the non-hanging pages. After calculating the PageRank, the hanging pages were added back without affecting the results. The authors claim that a couple of iterations are enough to remove most of the hanging pages. Kamvar et al. (2003) suggested by removing the hanging nodes and then re-inserting them for the last couple of iterations. Completely removing all the hanging pages will change the results on the non-hanging pages (Brin et al. 1998, Haveliwala 1999 and Kamvar et al. 2003) since the forward links from the pages are adjusted to consider the lack of links to unreferenced pages. They suggested that to jump to a randomly selected page with probability 1 from every hanging page. For example the nodes $V$ of the graph ($n = |V|$) can be partitioned into two subsets: (i) $S$ corresponds to a strongly connected subgraph ($|S| = m$) and (ii) The remaining nodes in the subset $D$ have links from $S$ but no forward links. There are also other researches (Lempel et al.2000 and Ng et al. 2001) proposed methods to handle hanging pages. They are different from us.

A fast two-stage algorithm (Pan-Chi Lee et al. 2003) for computing PageRank and its extensions are proposed based on the Markov chain reduction. The PageRank vector is considered as the limiting distribution of a homogeneous discrete-time Markov chain that transitions form 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 hanging pages are combined into one super node. In the second stage, the authors compute the limiting distribution of a chain where only the non-hanging pages are 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.

**PROPOSED METHODS**

We proposed two methods using a virtual node to handle the hanging pages in the graph. The Web is organized as a directed graph $G(V, E)$ with a vertex set of $V$ of $N$ pages and a directed edge set $E$. This directed graph is called as Web graph. The directed graph can be represented as matrix. PageRank creates graph and matrix before it computes the rank.

In method 1, a Virtual node ($V$) with self loop is connected and all the hanging pages are connected to the virtual node, a similar approach to Bianchini et al. (2005). In method 2, a Virtual node ($V$) with self loop is connected and all the pages including hanging and non-hanging pages are connected to the Virtual node $V$.

The basic PageRank model treats the whole Web as a directed graph $G(V, E)$, with a vertex set of $V$ of $N$ pages and a directed edge set $E$. The following matrix $M$ is an $m \times m$ matrix where $m = (n + 1)$ i.e. the last column and the last row is the virtual one which we are using in dealing with the hanging pages (Singh et al. 2010).

$$
M = \begin{bmatrix}
A_{1,1} & A_{1,2} & A_{1,3} & \cdots & A_{1,n} & A_{1,v} \\
A_{2,1} & A_{2,2} & A_{2,3} & \cdots & A_{2,n} & A_{2,v} \\
A_{3,1} & A_{3,2} & A_{3,3} & \cdots & A_{3,n} & A_{3,v} \\
A_{n,1} & A_{n,2} & A_{n,3} & \cdots & A_{n,n} & A_{n,v} \\
A_{h,1} & A_{h,2} & A_{h,3} & \cdots & A_{h,n} & A_{h,v}
\end{bmatrix}
$$

A sample directed graph with 6 nodes is shown in figure 1. There are 6 nodes in the directed graph. Nodes $A, B, D$ and $E$ are non-hanging pages (shown in blue color). Nodes $C$ and $F$ are hanging nodes.
(shown in red color) i.e. they do not have any forward links. Our study here is to show the effect of hanging nodes in the PageRank computation. Also it shows how the neighboring page ranks get affected by the hanging pages.

![Directed Web Graph with 6 nodes](image)

**Figure 1.** A Directed Web Graph with 6 nodes

### A. Markov Analysis

A.A. Markov, a famous Russian Mathematician invented Markov chain (Norris 1996) in the early 1900’s to predict the behavior of a system that moves from one state to another state by considering only the current state. Markov analysis can be defined as any system which uses the Markov chain to predict the probability of the future state by taking only the current state. Markov analysis is used in biology, economy, engineering, physics etc. Google search engine internally follows the Markov chain even though in the original PageRank paper (Brin et al., 1998) it was not mentioned but the other researchers (Langville et al. 2005, and Bianchini et al. 2005) proved that the PageRank algorithm of Google follows the Markov chain. Markov chain uses only a matrix and a vector to model and predict it.

PageRank model uses the random walk theory on the Web graph by moving from one node to another node randomly to compute the rank of a page. In this, some nodes are visited more often than others because those nodes are having more back links. So naturally they are important pages. When a hanging node comes, the random walk cannot proceed further, so it can only proceed further by other than moving from one node to another (user can type the URL on a Web browser). So the Stochastic interpretation of PageRank works only when there are no hangings pages (Bianchini et al, 2005). But in
reality there are lots of hanging pages on the Web (shown in the transition matrix below). So hanging pages cannot be ignored in PageRank calculation due to its importance.

**B. Transition Matrix**

The transition matrix $T$ for the above graph $G$ is shown as follows. It can also be called as hyperlink matrix. It is an $n \times n$ matrix, where $n$ is the number of Web pages. If Web page $i$ has $l_i \geq 1$ links to other Web pages and Web page $i$ links to Web page $j$, then the element in row $i$ and column $j$ of $T$ is $T_{ij} = 1/l_i$ where $l_i$ is the number of forward links of Web page $i$. Otherwise, $T_{ij} = 0$. Thus, $T_{ij}$ represents the likelihood that a random surfer will select a link from Web page $i$ to Web page $j$.

$$T_{ij} = \begin{cases} \frac{1}{l_i} & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The Transition Matrix $T$ can be produced by applying the equation (1) on the Web graph shown in the figure 1 is as follows:

$$T = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In the above Transition Matrix $T$, rows 3 and 6 are having only zeros. It means that node $C$ and node $F$ are hanging nodes and the probability is zero that a random surfer moves from node $C$ and $F$ to any other node in the directed graph. Matrix $T$ is not stochastic and it needs to be stochastic as per the PageRank model.
C. Method 1

Using our proposed method 1, i.e. connect a virtual node, \( V \) with self loop and connect all the hanging nodes the virtual node (shown in orange color), as shown in figure 2 and this matrix \( \tilde{T} \) is stochastic now.

![Directed Graph with virtual node V using method 1](image)

\[
\tilde{T} = \begin{bmatrix}
0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\
1/3 & 0 & 0 & 0 & 1/3 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 \\
0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

We applied the PageRank formula in equation (2) to calculate the page rank for the graph structure shown in figure 2. The sample calculation for the proposed two methods i.e. hanging pages connected to the virtual node and all the pages connected to the virtual node are shown below.

\[
PR(p) = \frac{d \sum_{q \in pa_p} \frac{PR_q}{O_q} + (1-d)}{1} + \sum_{q \in pa_p} \frac{PR_q}{O_q} + (1-d)
\]

Where \( p \) is an arbitrary page having back links from set of pages \( pa. O_q \) is the number of forward links of page \( q \) and \( d \) is the damping factor such that \( 0 < d < 1 \) and it is usually set to 0.85.
In the PageRank calculation, node A is getting back links from only nodes B and D. Only the forward links will be different for method 1 and method 2.

\[ PR(A) = d \left( \frac{PR(B)}{O(B)} + \frac{PR(D)}{O(D)} \right) + (1-d) \]  

We assumed the initial page ranks of all the nodes as 1 and the damping factor \( d \) is 0.85 and did the calculation. This calculation continues until the PageRank for all the nodes get converged. Later in the in the experimental section shows with the actual data from Web. Only a sample calculation is shown below:

\[ PR(A) = 0.85 \left( \frac{1}{3} + \frac{1}{3} \right) + (1 - 0.85) = 0.717 \]  

**D. Method 2**

In method 2, a Virtual node with self loop is connected and all the pages including hanging pages and non hanging pages are connected to the virtual node. The directed Web graph shown in figure 1 is modified using method 2 and is shown in figure 3 as follows:

![Directed Graph with virtual node V using method 2](image)

**Figure 3.** Directed Graph with virtual node V using method 2

The transition matrix for the method 2 is shown below:
In the transition matrix, the last column i.e. the virtual node has more transition probability because every node is connected to it. This makes the virtual node PageRank goes very high and that will not be presented to the user.

The same PageRank formula shown in equation (2) is applied to the directed Web graph on figure 3 using method 2. Here, every node gets an additional forward link because all the nodes are connected to the virtual node. In this method the PageRank values are reduced for all the nodes but the virtual node gets more forward links and its PageRank score goes high. In the final ranking order, the virtual node will not be shown. The PageRank of $A$ goes down from 0.717 to 0.5325 due to the rank distribution and the rank goes down uniformly for all the pages and it does not affect the order.

$$PR(A) = 0.85 \left( \frac{1}{4} + \frac{1}{5} \right) + (1 - 0.85) = 0.5325 \quad (5)$$

The above equations (4) and (5) are the first iteration of the PageRank computation for method 1 and method 2 and the PageRank will get converged after so many iterations.

E. TrustRank Algorithm

TrustRank (Gyöngyi et al. 2004) is one of the popular link based Web spam detection algorithm which works closely with PageRank algorithm. Web Spam (Gyöngyi et al., 2005) refers to the sites/pages created with the intention of misleading the search engines. Relevancy and quality are the two important factors for search engines when they process and produce search results. Some sites or pages use various techniques to achieve higher-than deserved ranks is called as Web spamming or spamdexing (Gyöngyi et
TrustRank algorithm separates good sites from spam sites using semi-automated methods by assuming that good sites seldom points to spam or bad sites. TrustRank works by selecting good seed set. To select good seed set, it uses inverse PageRank and uses the link structure of the Web to flow the trust from good pages to other good pages and separates all the good pages for the seed set. Then it sorts the results in descending order to select top $n$ good pages as a seed set. Then the TrustRank normalize the distribution vector by applying on the following equation (6).

$$t^* = \alpha T \cdot t^* + (1 - \alpha) d$$

(6)

Where $\alpha$ is the decay or damping factor is normally set to 0.85, $T$ is the transition matrix, $d$ is the distribution vector after normalization and $t^*$ is the TrustRank scores. It is an iterative algorithm like PageRank and gets converged in $M$ iterations.

**EXPERIMENTAL RESULTS**

The PageRank and the TrustRank program are implemented in Python program and tested on an Intel Core 2 (2.40 GHz) with 4GB RAM.

A. Dataset

The dataset used in the experiments is provided by the European Archive Foundation, with the support of the LiWA - Living Web Archives project known as EU2010 collection (András et al. 2010). In this experiment, we used hostgraph instead of webgraph due to the large collection of the dataset. The original webgraph contains 23m Web pages while the hostgraph contains 191388 different hosts and 103749 hanging hosts (host that are not pointing to other hosts) as shown in Figure 4. In this experiment, we show the rank of all hanging hosts for method 1 and method 2. Furthermore, we applied the famous Web Spam detection algorithm, TrustRank (Gyöngyi et al. 2004) on the same dataset and compared with the results from TrustRank with virtual node.
Sample Dataset types

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of Hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hanging</td>
<td>103749</td>
</tr>
<tr>
<td>Non-hanging</td>
<td>87639</td>
</tr>
<tr>
<td>Total</td>
<td>191388</td>
</tr>
</tbody>
</table>

![Distribution of hanging and non-hanging hosts](image)

**Figure 4.** Distribution of hanging and non-hanging hosts

### B. Pseudo Code

The following Pseudo code is same as the PageRank algorithm. We implemented this Pseudo Code using our proposed method 1 and method 2.

```
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)
```
C. Experiments

For the experiments, we used the damping factor $d$ of 0.85 and run in 50 iterations. We are confident that 50 iterations are enough to achieve convergence.

![Figure 5. Ranking results of hanging Host for Method 1](image)

In method 1, we included a virtual node with self loop and connect all the hanging hosts to the virtual node. Figure 5 shows the rank results of hanging host for method 1. The Y axis denotes the rank values of the hanging pages and the X axis denotes the 1$^{\text{st}}$ hanging node until the 103749$^{\text{th}}$ hanging node (the last node).

![Figure 6. Ranking results of hanging host for method 2](image)
In method 2, all the nodes are connected to the virtual node to make the forward link uniform for ranking purpose. Figure 6 shows the rank results of the hanging host for method 2.

**Figure 7. Ranking Results from TrustRank**

**Figure 8. Ranking Results from TrustRank with Virtual Node**
In figure 7, it shows the ranking results on the good sites on EU2010 using TrustRank and in figure 8, it shows the ranking results on the good sites on EU2010 with virtual node. We took a sample of 1309 good sites provided by the dataset to see the difference on TrustRank and TrustRank with virtual node.

D. Result Analysis

Method 1

In the experimental method 1, we included a virtual node with self loop and connected all the hanging hosts to the virtual node. A graph is generated and given to the program for computation of page ranks. The page rank results are shown in figure 5. Here, the page ranks are fair by including all the hanging hosts in the ranking but the number of iterations is more. The output is shown in figure 5 as a chart format by showing the rank values on the Y axis and the order of hanging hosts on the X axis.

Method 2

In the experimental method 2, all the hanging hosts as well as the non hanging hosts are connected to the virtual node to make the out link uniform for the ranking purpose. The output is shown in figure 6 in a chart format. The number of iteration is less here comparing to method 1. It also produces fair and accurate ranking of hosts. The page rank value reduces a little bit here compare with method 1 because the forward links of all the hosts are connected to the virtual node. The original PageRank method is not good as for the hanging pages are concern because they are omitted in the computation. In method 1, the computation includes the hanging hosts but the number of iteration is more. Our experimental result proves that method 2 is better by not only reducing the number of iterations in the computation but also produces a fair and accurate ranking of results. The output is shown in the chart format in figure 6. Both our proposed methods include hanging hosts in the ranking process where the original PageRank method does not include the hanging pages in the ranking process which makes our methods produces more relevant and accurate ranking results. It is very clear that there are more hanging pages on the Web (54%
are hanging pages in our sample data set) than the non-hanging pages in the Web and the rate is keep increasing. So the hanging pages cannot be neglected in ranking process due to their importance.

In figure 7 and figure 8, there is no significant difference with/without virtual node in the calculation of TrustRank. Our proposed methods are capable of combating Web spam with the inclusion of TrustRank.

E. Computation Complexity

The basic PageRank model treats the whole web as a directed graph $G = (V, E)$, with a set $V$ of vertices consists of $n$ pages, and the set $E$ of directed edges $(i, j)$ which exist if and only if page $i$ has a hyperlink to page $j$. The directed graph can be represented as an $n \times n$ matrix.

According to (Augeri 2008, Safronov et al. 2002) the PageRank algorithm for every Web page, we need to find all pages that the new page links to. This requires a full array of scan so that every element needs to be checked. If $n$ is the number of web pages, we can safely make an assumption that $n-1$ is the maximum number of links on a page. So therefore the worst case performance is $O(n^2)$. The PageRank algorithm which is essentially a power method algorithm has a lower and upper bound of $\Omega(n^2 \log n)$ and $O(n^2 \cdot t)$ where $n = |V|$, $t = \log_d \tau$, $d$ is a damping factor, usually 0.85 and $\tau = 1/n$. If sparse matrices are used, the lower and upper bound of the PageRank algorithm are $\Omega(m^2 \log n)$ and $O(m \log d \tau)$ respectively, where $m$ denotes the number of edges contained in the graph.

Our proposed method for handling hanging pages is by adding a virtual node into the calculation. Virtual node can be denoted by $\varepsilon_v$, so the lower and upper bound of $\Omega(n^2 \log n)$ and $O(n^2 \cdot t)$ where $n = |V + \varepsilon_v|$, $t = \log_d \tau$, $d$ is a scaling factor, usually 0.85 and $\tau = 1/n$, $d$. It would not be affecting the PageRank algorithm, and by adding the virtual node, it is actually taking all the hanging pages into account. Intuitively, our proposed algorithm has the same computation power with PageRank algorithm and also produced results as our proposed method that takes hanging pages into consideration.
CONCLUSION

This paper proposes two methods using virtual node to calculate PageRank in dealing with hanging pages and the experimental results are shown. Our proposed method 2 takes less number of iterations and also it produces fair and accurate ranking of pages. Most of the Web ranking algorithms are kept as trade secret due to competition. So it is difficult to know how the ranking algorithms are implemented in real but with our limited resources, we implemented a page rank algorithm and handled the hanging pages with the best of our knowledge. We also implemented TrustRank algorithm to combat spamming in our proposed methods. Our two proposed methods (method 1 and method 2) are compared with the standard PageRank algorithm. The future work could be in the direction of improving the computation complexity.

REFERENCES


This paper is an accepted manuscript in Cybernetics and Systems An International Journal.

The original published manuscript can be found in:


Or DOI: 10.1080/01969722.2011.634679