UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

# AN ANALYSIS OF LINK BASED WEB SEARCH ALGORITHM

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

ZILIANG JIAO
Norman, Oklahoma
2015

AN ANALYSIS OF LINK BASED WEB SEARCH ALGORITHMS

A THESIS APPROVED FOR THE SCHOOL OF COMPUTER SCIENCE

BY

# Acknowledgement

First, I would like to express my appreciation to my advisor, Dr. Lakshmivarahan. I am gratitude for the opportunity that he gives me to do research and further study with him. I am thankful for his instruction and help. His guidance, encouragement and assist are the key to the accomplishment of this thesis. I have learned a lot from his during the period of this work.

Second, I also want to give my appreciation to the members of my thesis committee, Dr. Radhakrishnan and Dr. Dhall. I thank them for offering their valuable time to serve on the committee.

Finally, I am greatly thankful to my parents and family members. Without their support and caring, it is impossible for me to complete anything that I achieved. They are the reason that I keep studying and conquering the difficulties.

# Table of Contents

Acknowledgement ........ iv
Table of Contents ........ v
Abstract ........ vi
Chapter 1 Introduction ........ 1
Chapter 2 Web Search Structure ........ 6
    2.1 The Structure of Web ........ 6
    2.2 The Structure of Search Engine ........ 11
Chapter 3 PageRank Algorithm ........ 15
    3.1 PageRank Introduction ........ 15
    3.2 The Original Idea ........ 15
    3.2 Google Matrix ........ 18
    3.3 Random Walk and Markov Chain ........ 20
    3.4 Compute The PageRank Vector ........ 22
    3.5 Convergence of PageRank ........ 24
    3.6 Mathematical Properties of PageRank ........ 24
        3.6.1 Linear System ........ 25
        3.6.2 Sensitivity of PageRank ........ 26
            3.6.2.1 Sensitivity with respect to α ........ 26
            3.6.2.2 Sensitivity with respect to S ........ 27
            3.6.2.3 Sensitivity with respect to v ........ 29
    3.7 Personalized PageRank ........ 30
    3.8 Performance Evaluation ........ 32
    3.9 Conclusion ........ 36
Chapter 4 HITS Algorithm ........ 37
    4.1 HITS Introduction ........ 37
    4.2 The Original HITS Algorithm ........ 38
    4.3 The Computation of HITS ........ 39
    4.4 HITS Stability ........ 42
    4.5 HITS Convergence ........ 45
    4.6 Performance Evaluation ........ 46
    4.7 Conclusion ........ 53
Chapter 5 SALSA Algorithm ........ 54
    5.1 SALSA Introduction ........ 54
    5.2 SALSA Algorithm ........ 55
        5.2.1 SALSA Structure ........ 55
        5.2.2 Random Walk on Bipartite Graph ........ 57
    5.3 SALSA Computation ........ 58
    5.4 Markov Chains with Multiple Irreducible Components ........ 59
    5.5 SALSA on Irregular Graphs ........ 61
    5.6 Performance Evaluation ........ 63
    5.7 Conclusion ........ 68
Chapter 6 Conclusion ........ 70
References ........ 71
Appendix A ........ 77

# Abstract

With the growth in the past decades, web search has shown its dominating role on the internet. Researchers have been studying different type of approaches to make searching more effective and faster. There are some basic algorithms used by various search engines. Here we focus on three main algorithms. They all obtain the common concept called link analysis. The widely known PageRank algorithm formed the theory of web search and is used by Google search engine. Hyperlink Induced Topic Search (HITS) algorithm gives a different perspective of link analysis in terms of finding authorities according to the query. "Stochastic Approach for Link-Structure Analysis (SALSA) shows the combination of the other two and has its application in some specific area. These ideas have common characteristics, yet they analyze the huge web in different ways and have their own uniqueness. We compare the advantages and disadvantages of these algorithms base on their mathematical properties and performances.

# Chapter 1

# Introduction

Ever since the web has played a dominating role in contemporary life, the concept of information retrieval has been elevated toward a different standard. We start to look for a handy way to search things that we need. Therefore, it is necessary to have an analysis of current web searching technique and figure out whether there is any improvement we can make.

Information retrieval (IR) [Singhal 2001] is the process of searching among a collection of documents and acquire a particular piece of information. With the invention of internet, the World Wide Web became a signal of Information age and web search grew into a revolutionary stage. IR system locates information that is relevant to users' search purpose. The growing need of an IR system is similar to Moore's law.

The biggest jump of searching approach happened in 1998 when link analysis [Borodin et. al. 2005] and searching of anchor text hit the information retrieval scene. The most successful search engines are using such technique to exploit information inherent in the structure of the web. Almost all major search engine techniques depend on link analysis and combined some traditional scoring approaches.

Search engines started to grow and widely used in the late 1990s. The first tool used for searching on the Internet was Archie. It was created in 1990 by Alan Emtage, Bill Heelan and J. Peter Deutsch in Montreal. The program downloaded the directory listings of all the files located on public anonymous FTP (File Transfer

Protocol) sites, creating a searchable database of file names.

In June 1993, Matthew Gray, then at MIT, produced what was probably the first web robot, the Perl-based World Wide Web Wanderer, and used it to generate an index called 'Wandex'. The web's second search engine Aliweb appeared in November 1993. Ali web depended on being notified by website administrators of the existence at each site of an index file in a particular format.

JumpStation, created in December 1993, used a web robot to find web pages and to build its index, and used a web form as the interface to its query program. It was thus the first WWW resource-discovery tool to combine the three essential features of a web search engine (crawling, indexing, and searching). Because of the limited resources available on the platform it ran on, its indexing and hence searching were limited to the titles and headings found in the web pages the crawler encountered.

One of the first "all text" crawler-based search engines was WebCrawler, and it came out in 1994. It allowed users to search for any word in any web page, which has become the standard for all major search engines.

In 1996, Netscape was looking to give a single search engine an exclusive deal as the featured search engine on Netscape's web browser. There was so much interest that instead Netscape struck deals with five of the major search engines, each search engine would be in rotation on the Netscape search engine page.

Google adopted the idea of selling search terms in 1998, from a small search engine company named goto.com. Around 2000, they achieved better results for many searches with an innovation called PageRank, as was explained in the paper

*Anatomy of a Search Engine* written by Sergey Brin and Larry Page. This algorithm ranks web pages based on the link analysis of web sites and pages that have link between each other, the core idea of this algorithm is that good pages are linked to more than others.

Many search engine companies were caught up in a speculation-driven market boom that peaked in 1999. By 2000, Yahoo! was providing search services based on Inktomi's search engine. Microsoft first launched MSN Search in the fall of 1998 using search results from Inktomi. In 2004, Microsoft began a transition to its own search technology, powered by its own web crawler.

In the span of a decade, the World Wide Web has grown from a small research project into a vast repository of information and a new medium of communication. Unlike other great networks of the past century, the web is a network of content and hyperlinks, with over a billion interlinked “pages” created by the uncoordinated actions of tens of millions of individuals. Because of the decentralized nature of its growth, the web has been widely believed to lack structure and organization. Recent research, however, shows a great deal of self-organization. Thus, a lot of analyses of the Web grape are based on hyperlinks which have revealed an intricate structure that is proving to be valuable and crucial for organizing information, This has improved the understanding of search methods and social context. The Web contains a large, strongly connected core in which every page can reach every other by a path of hyperlinks. This core contains most of the prominent sites on the Web. The remaining pages can be characterized by their relation to the core.

Before the development of link analysis, there are a series of changes happened to the search engine throughout the history. Among all the searching techniques, three models mainly applied are, Boolean search engine, Vector Space model and Probabilistic model [Baeza-Yates et. al. 1999]. In this category, some algorithm has been studied for a long while, this includes TF-IDF [Salton et. al. 1975] and BM25 [Zaragoza et. al. 2004], which are known as state of art text feature that can usually achieve a very good performance when used to find rich documents. But they all have distinct drawbacks. The web is huge and dynamic. There are billions of pages and they are change from time to time. New pages will be added and old pages could be deleted. Meanwhile, the content of these web pages can also be different every day. In this way, the analysis of hyperlink will provide a great leap to the study of search engine.

The link analysis idea have played an integral role in the ranking functions of the current generation of Web search engines, including Google, Yahoo!, Microsoft's search engine Bing, and Ask. Most people do not know much about what to do with the huge web until the introduction of PageRank and HITS algorithm. These two techniques brought the understanding of hyperlinks and recommendations. In the late 1990s, it was possible to produce reasonable rankings using these link analysis methods almost directly on top of conventional search techniques; but with the growth and enormously expanding diversity of web content since then, link analysis ideas have been extended and generalized considerably, so that they are now used in a wide range of different ways inside the ranking functions of modern search engines. There are a few algorithms on the mainstream today and being applied to many search engines. The most popular ones are

Google’s PageRank algorithm and Kleinberg’s HITS algorithm. After they have been developed, there are some derivatives and improvements made based on these two techniques. One of the algorithm that has an crucial effect is the SALSA algorithm. There are also a lot of variations, but their core idea come from those three original web search algorithms. In summary, many algorithm still keep the similar structure to do link analysis. Therefore, we want to focus on the original property of these algorithm and show people the reason that they achieved great success and dominated the information era.

In this thesis, we are going to describe these algorithms and we will show you some of the difference in terms of performances and complexity. During the analysis of these algorithms, we will also mention how much the effect is whether the ranking is determined on-line or off-line.

# Chapter 2

# Web Search Structure

## 2.1 The Structure of Web

The web contains billions of pages together with links between them. These links create paths from one page to another, and allow people to go from one site to almost every other. Although the web is very rich, it creates a huge challenge for people to retrieve useful information. For people who tried to design a search engine for such information retrieval, the challenge is even higher. The web does not have a designed structure to follow, it consists of text, hyperlinks and network. Due to the growth rate of information, web structure becomes more and more intricate. How to organize, understand and utilize the structure in order to improve searching technique is crucial.

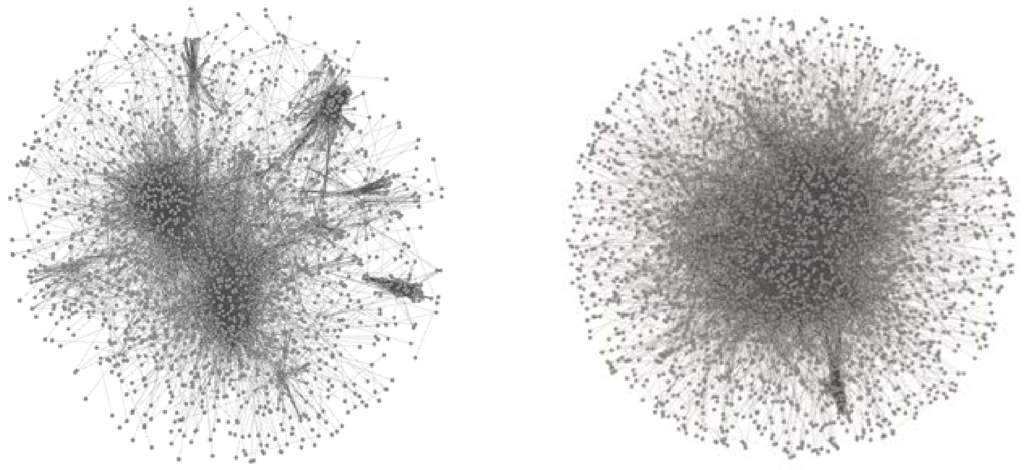


Figure 2.1 Maps of subset of the Web

The web is consist of a variety of network structures [Newman 2003]. A social network is a set of people or groups of people with some resource of contacts or interactions between them. The classic example of an information network is the network of citations between academic papers. Most learned articles cite previous work by others on related topics. These citations form a network in which the vertexes are articles and a directed edge from A to article B indicates that A cites B. Technological networks are typically designed for the study of power and water flow. Relevant corporations need to study delivery system such as the telephone networks. A number of biological systems can be usefully represented as networks. Some classic examples are a biological network, genetic network and food network. Neural network has also been widely studied as a pattern of intelligence system.

Viewing these networks in terms of their graph structures provides significant insights, and the same is applied to the Web. When we view the Web as a graph, it allows us to better understand the logical relationships expressed by its links; to break its structure into smaller, cohesive units.

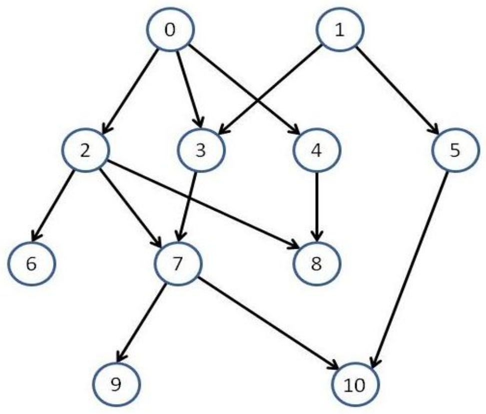


Figure 2.2 Directed graph representing the web

Some researches have shown that the in-degree and out-degree of an individual page follows the power law distribution [Kleinberg et. al. 2001], as shown in Figure 2.3, which gives a plausible idea of how can we possibly refine the web graph and extract the core structure for our analysis. To some extent, it indicates that with the growth of network, the probability the existing node get new links is proportional to its in-degree. Moreover, the distribution of visitors on the web page generally follows Zipf's Law, which tells us that a small number of sites such as Yahoo are extremely popular and captured a disproportionate amount of traffic. Figure 2.4 shows such a plot for the same data set of online users' site visits. In general, it tells us that the most popular sites would have to be more popular, and the less popular sites slightly more numerous.

Such observation leads to a useful analysis of the web. The page with high density usually indicate that they are strongly related to the query topic. This provides an important condition for HITS algorithm.

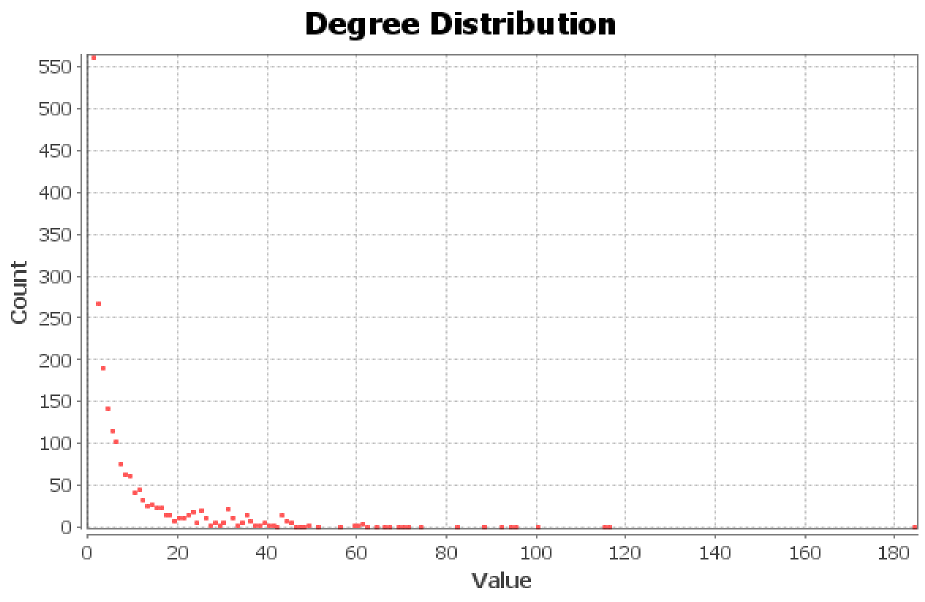


Figure 2.3 Degree Distribution of a Web Graph

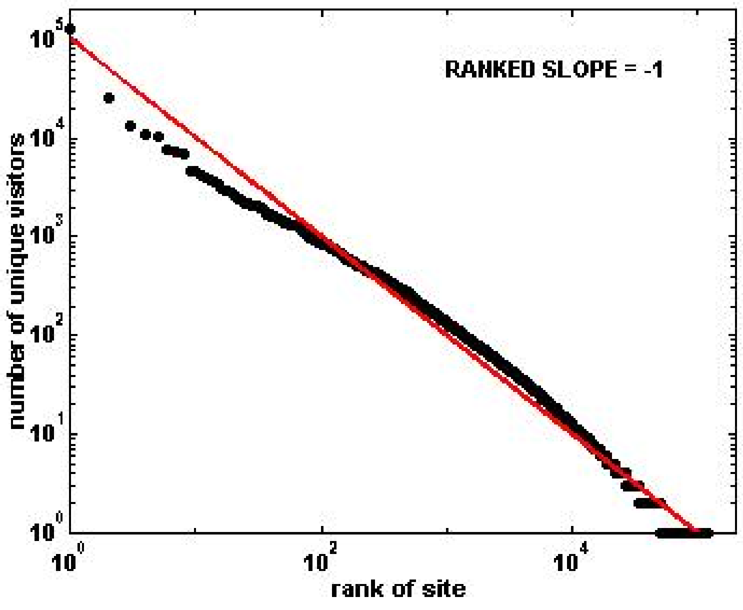


Figure 2.4 Site Rank versus their popularity

One other interesting property about the web graph is strongly connected components. By mining the strongly connected components, we find that 48% of the web pages are in the same component with other pages scattered in other components.

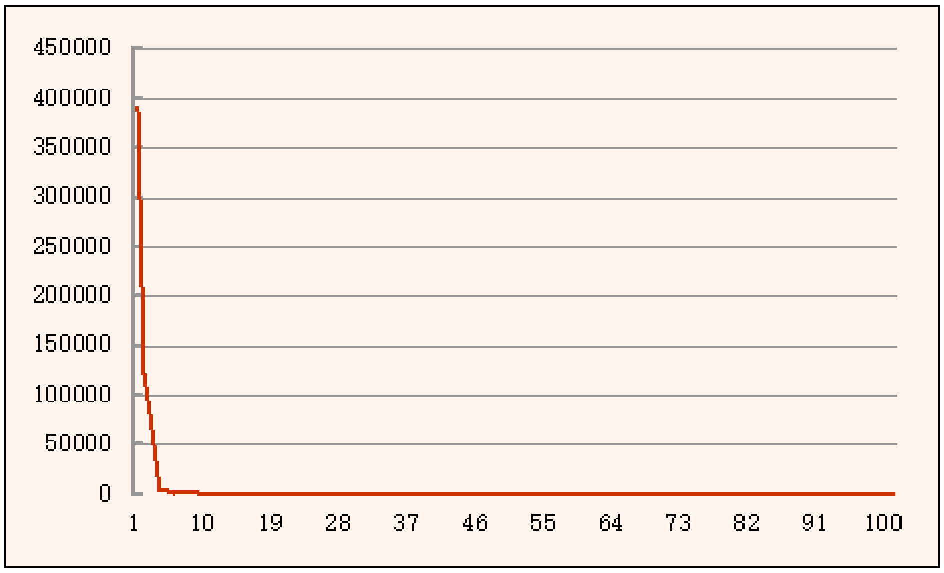


Figure 2.5 Distribute of Strongly Connected Components on Sample Web

This leads to a observation of a very common web structure called bow-tie structure proposed by [Broder et. al. 2000].

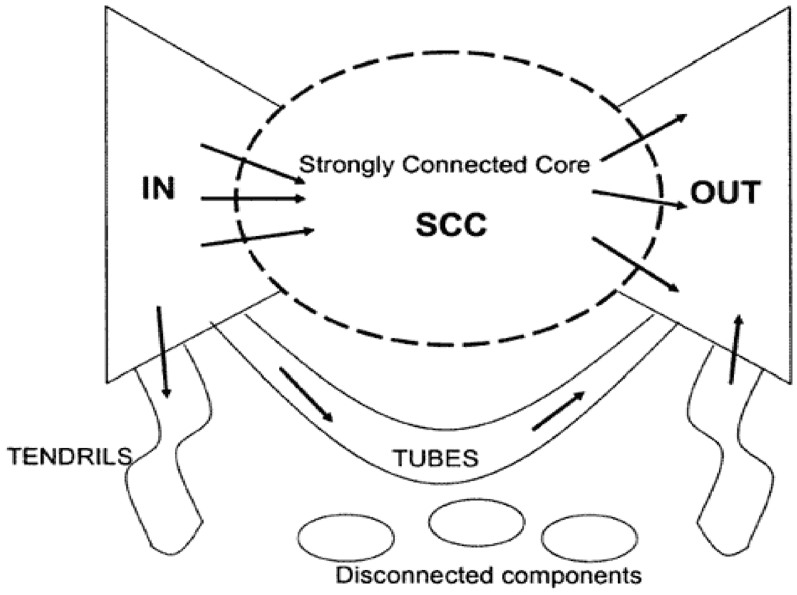


Figure 2.6 A schematic picture of the bow-structure of the Web the structure has persisted despite the number of nodes

Take it as a whole, the bow-tie picture provides a overview of the Web's structure, based on its reachability properties and how its strongly connected components fit together. From it, we see that the Web contains a central "core" containing many of its most prominent pages, with many other nodes that lie upstream, downstream, or "off to the side" relative to this core.

Link analysis has become a major approach for organizing and retrieving information on the Web. It gives a comprehensive analysis of relations between pages and a means of finding authoritative, relevant sources on the Web. It has proven to be useful in the design of modern day search engines.

Google PageRank [Brin et. al. 1998] models a web page's authoritativeness by the number of links pointing to it from other pages by recursively factoring in the referring page’s authoritativeness. The number of incoming links follows a power-law distribution wherein few pages get most of the links, and the pages are clustered such that a core of well-connected pages is separated by paths of fewer than 20 hops. HITS [Kleinberg 1998], in other way, cluster pages into a community that contains most relevant pages. The pattern in such community consists of a collection of hub pages that are resources that guide authorities, which is a collection of pages that are strongly related to the topic. SALSA [Lempel et. al. 2000], which combines the properties from PageRank and HITS, aims at making algorithm to be computationally efficient and solving some problems from the original approaches.

Analysis of the Web’s structure is leading to improved methods for accessing and understanding the available information, for example, through the design of better search engines, automatically compiled directories, focused search services, and content filtering tools.

## 2.2 The Structure of Search Engine

Web search is a composition of hardware and software process. The main components of web search process contains crawling and indexing. Every time the user types a query on the search engine, it will process the query, find the appropriate web pages from the web server and return it back to the user. Figure 2.5 shows a brief structure of modern day search engines.

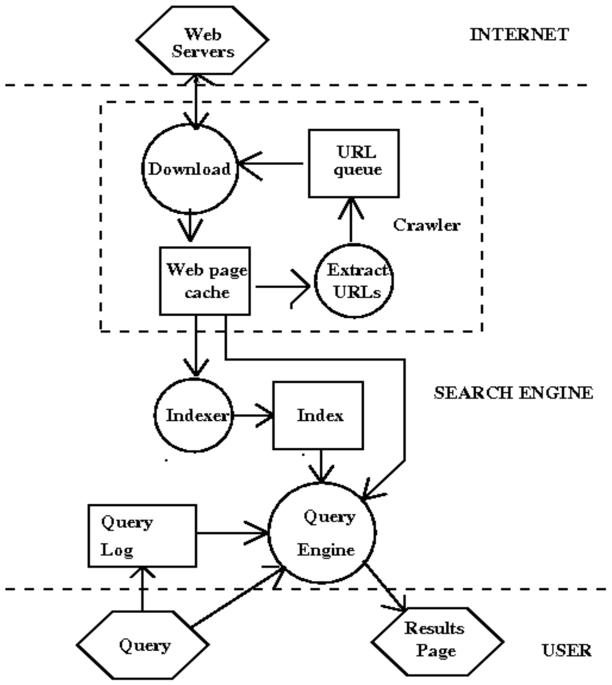


Figure 2.7 Search Engine Structure

Web search engine has a crawler module. Crawling software create Spiders that explore the web to gather information and store them in a central repository. These information will be sent to the index module which take s each page and extracts vital descriptors. Indexes hold information for each web page. They formed a module called query-independent. Crawling and Index are the basic elements in search engine. They seems to be less attractive but also important. The crawler contains a short program that instructs machine on how and which page to retrieve. The crawling module gives a root set of URLs and tells the path from one page to another. Crawlers must carefully select pages because sometimes search engine only focus on a specific portion of the web. Once new page that a spider brings back is sent to the index module, software programs parse the content and take valuable information. That piece of information is contained in title, description, and anchor text. The content index stores textual information and the inverted file is used to store compressed information.

A search engine is a system run by programs which “crawl” the web for information including web pages, images, files, videos and other documents. When a new document is detected, its data is stored on the store-server in a process referred to as caching. The documents in the search engine’s cache are then ranked in order of importance on its Search Engine Result Page Crawling refers to the ability of a search engine to traverse the billions of interlinked pages on the World Wide Web. With massive amounts of pages being generated on an hourly basis, it is impossible for us humans alone to visit, record and organize them on our own. Instead, automated search crawlers or “bots” conduct regular searches to save us the agony of finding relevant content ourselves.

Search bots wait upon signals from previously indexed pages, such as links, to be notified of new content. So if you have created a new page on your website and linked to it from an existing page or the main menu, this would be a signal to search bots that they should come visit and index it.

After a web page or document has been detected by crawlers, all its accessible data is stored on search engine indexing servers so it can be retrieved when a user performs a search query. Indexing serves two purposes, to return results related to a search engine user’s query, and to rank those results in order of importance and relevancy. The order of ranking is dependent with each search engine’s ranking algorithm.

Search engines have programs that crawl the World Wide Web through links and other means to find new and updated web content. Once found, each web page is processed using an algorithm and indexed. When a search user performs a query, the indexed result is retrieved and ranked in order of relevancy to the users query. The query module depends on users. A query is typically only an approximation of the user’s intention, and the same query may express many different intentions. Content score computed from the index and inverted file is called query-dependent. Content score computed from the structure index is query-independent. Thus, popularity score can be measured in different manner and it largely depend on how people design search engines. However, one most important aspect of web search is how the ranking algorithm always provide the information that people are more likely to explore. So what we want to do is that we would like to look deep inside these popular ranking algorithms and give a comprehensive analysis of these algorithms.

In the following chapters, we will concentrate on the algorithms that bring us these results of ranking in the search engine and we will focus on how exactly they give us the rank.

# Chapter 3

# PageRank Algorithm

## 3.1 PageRank Introduction

Every time we search on this web, we want the most relevant pages to be returned according to each query. These pages need to be ranked so that we can find pages contains contents close enough to the topic we want. To measure the web pages and return search results based on their importance, PageRank means to compute the ranking of all pages on the web graph. It is an algorithm used by Google Search to rank websites in their search engine results. According to Google: PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites. So the intuition of PageRank is to estimate the importance of pages through the link structure. We describe it as a page has higher rank if the sum of the score of the nodes pointing to it is higher. It starts with simple weight-sharing based on in-links, and refines it with the iteration of its scores.

PageRank is used as a network centrality measure. A network centrality score yields the importance of each node in light of the entire graph structure. PageRank also means to illuminate a region of a large graph around a target set of interest. It is also called personalized PageRank based on PageRank's origins in the web.

## 3.2 The Original Idea

The computation process of PageRank can be derived this way. Whenever a user clicks the hyperlink of the current page, he or she is following the one of the outgoing links of this page and jump into another one. We assume page A is pointed by $P_1$...$P_n$. There is a scaling factor c to be set between 0 and 1. PR(P) is the PageRank and N(P) is the number of links going out of P. The PageRank of one page is defined as follows,

$$PR(P) = PR(P_1) / N(P_1) + \cdots + PR(P_n) / N(P_n) \qquad (3.1.1)$$

Based on the discussion and formula above, let us start with a simplified version of PageRank calculation. Suppose that $\left|B_{P_j}\right|$ is a set of links pointing to $P_i$, $P_j$ is in $\left|B_{P_j}\right|$, and $\left|P_j\right|$ is the number of out-links from $P_j$. Then the rank of a page $P_i$ at one point of iteration is given by

$$r_{k+1}(P_i) = \sum_{P_j \in B_{P_j}} \frac{r_k(Pi)}{|Pj|} \qquad (3.1.2)$$

In order to compute the score of each page, we can assume web graph with n nodes, each nodes have in-links and out-links. The probability of this clicking manner on any of the links will be equal. So the score of all nodes are initially set to be 1/n. Then it performs a sequence of updates. Each page divides its current score equally across its out-going links, and passes these values to the pages it points to. Each page updates its new PageRank to be the sum of the shares it receives. Applying this equation to the web in Figure 3.1, we can have a sample result of the PageRank after a few iterations.

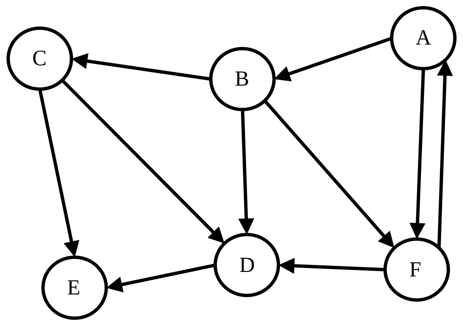


Figure 3.1 A Sample Directed Graph

Table 3.1 Few iteration steps performed on Figure 3.1

| Step | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | 1/6 | 1/6 | 1/6 | 1/6 | 1/6 | 1/6 |
| 2 | 1/12 | 1/12 | 2/9 | 2/9 | 1/4 | 5/36 |
| 3 | 5/72 | 1/24 | 5/18 | 15/72 | 1/3 | 5/72 |

This rank corresponds to the standard probability distribution of random walk. However, there is a problem of this formula. Consider a few pages pointing to each other and do not have a link pointing to other pages, the score of these pages will as a result be accumulated more after each iteration and do not share at all. Therefore the phenomenon like that causes a rank sink. One other problem is that if two pages form a cycle, then the iteration will never converge throughout the process as the score will flip between 0 and 1. To overcome this phenomenon, Brin and Page introduced a source of rank. It is a vector over the web pages which is used as a source to make up for the rank sinks or cycles. This can be referred to as a random surfer model. Imagine someone who is randomly browsing a set of web pages. They start by choosing a page with equal probability. Then follow links for a sequence of steps and follow it to where it leads. Such an exploration of nodes performed by randomly following links is called a random walk on the network. This means that when he arrives at one page, he could choose a hyperlink at random with probability α or to any other sites by the distribution E at the probability 1-α .This gives the standard summation formula.

$$r_{k+1}(P_i) = \alpha \sum_{P_j \in B_{P_j}} \frac{r_k(P_j)}{|P_j|} + (1-\alpha)E \qquad (3.1.3)$$

During the computation. The algorithm is initiated with $r_0(Pi) = 1/n$ for all pages, then the process is repeated with iteration so that the PageRank score will converge to a stable value. Thus, we will have a conclusion that a node with more in-links will have a higher rank.

## 3.2 Google Matrix

We now describe an equivalent definition of PageRank that looks different on the surface, but in fact expresses the same idea. It works as follows.

We are going to present a comprehensive analysis of the Google Matrix. Given a web graph G=(V, E), we will analyze the link between nodes based on the simplified version. In order to generate the Google matrix G, we must first generate an adjacency matrix A which represents the relations between pages or nodes,

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

Assuming that a web user is equally likely to choose any of the out-links to go from one page to another, we can construct a matrix H from P

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

Some rows of the matrix, such as row 5 in this example, contain all zeroes. H is not stochastic. This occurs whenever a node contains no out-links. Many such nodes exist on the web. They are called dangling nodes. One remedy as Brin and Page suggested is to replace all zero rows 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 matrix H becomes a stochastic matrix S, the entry of the matrix H is the representation of Markov Chain

$$S = \begin{bmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \end{bmatrix}.$$

Now S is stochastic, but not primitive. To guarantee its convergence, Brin and Page applied an adjustment. As is described, a person can either follow the hyperlink of pages when he or she is browsing, or randomly enters a new page at random time. The representation of the concept of random walk on the the web graph is given by Google matrix. The random surfer model is given by $E = \frac{1}{n}ee^T$ meaning the probability of random surfing is equally likely. This results in the model of Google matrix G.

$$G = \alpha S + (1-\alpha)1/nee^T \tag{3.2.1}$$

In this example, when we take α as 0.9, the matrix G is

$$G = \begin{bmatrix} 1/60 & 7/15 & 1/60 & 1/60 & 1/60 & 7/15 \\ 1/60 & 1/60 & 19/60 & 19/60 & 1/60 & 19/60 \\ 1/60 & 1/60 & 1/60 & 7/15 & 7/15 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 11/12 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 1/60 & 1/60 \\ 7/15 & 1/60 & 1/60 & 7/15 & 1/60 & 1/60 \end{bmatrix}$$

Every node is now directly connected to each other, which makes G irreducible. Although the probability of transitioning may be very small in some cases, it is always nonzero. The irreducibility adjustment also insures that G is primitive, which implies that the power method will converge to the stationary PageRank vector $\pi^T$. As mentioned above, the pagerank is given by iteration of summations. This matrix is next applied to the computation of the PageRank vector.

## 3.3 Random Walk and Markov Chain

We start the computation of page rank with a stochastic matrix S. When users randomly follow the links to visit next page, the ranking vector is assigned by a given probability distribution. Therefore, we can have a conclusion that PageRank vector can be defined as a stationary probability after a long term random surfing. A random walk on a directed graph is nothing but a Markov chain. Given a graph and a starting node, we select a neighbor of it at random, and move to this neighbor; then we select a neighbor of this node and move to it, and so on; the sequence of nodes selected this way is a random walk on the graph.We denote this probability vector as the stationary vector for Markov Chain.

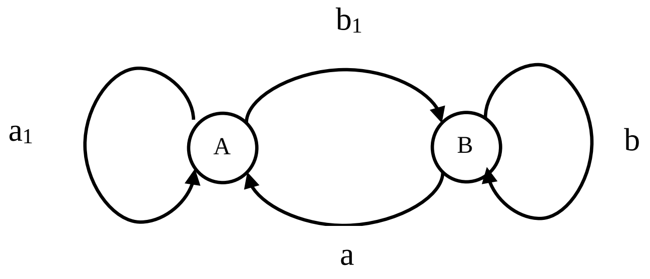


Figure 3.2 A directed graph with transition probability

A Markov chain is a sequence of random variables $X_0$, $X_1$, ... , $X_k$ that obey the *Markovian property,* that is the conditional probability distribution of future states depends only upon the one state ahead, not on the sequence of events that preceded it. A random walk is mostly represented by its transition matrix P. P is a square matrix denoting the probability of transitioning from any vertex in the graph to any other vertex. The transition probability is given by

$$P( X_{t+1} \mid X_t ) = \Pr ( X_{t+1} = Y \mid X_t = Y_t ) \qquad (3.3.1)$$

In figure 3.1, it is a directed graph with some probability of state changes. The transition matrix P is $\begin{bmatrix} a_1 & a_2 \\ b_2 & b_1 \end{bmatrix}$. Consider the state transition in Figure 3.2, with the transition matrix $P = \begin{bmatrix} a_1 & a_2 \\ b_2 & b_1 \end{bmatrix}$, where $a_1+a_2=1$ and $b_1+b_2=1$.and initial state $\pi(0) = \begin{bmatrix} \pi_1(0) \\ \pi_2(0) \end{bmatrix}$, if we have a distribution π over the nodes, we can obtain the distribution after one step by computing $\pi=P^T\pi$,

$$\pi(1)=\begin{bmatrix} a_1 & a_2 \\ b_2 & b_1 \end{bmatrix}\begin{bmatrix} \pi_1(0) \\ \pi_2(0) \end{bmatrix}=\begin{bmatrix} a_1\cdot\pi_1(0)+a_2\cdot\pi_2(0) \\ b_2\cdot\pi_1(0)+b_1\cdot\pi_2(0) \end{bmatrix}$$

Follow this computation rule, we have a iteration process,

$$\pi(1)=P^T\pi(0)$$

$$\pi(2)=P^T\pi(1)=P^TP^T\pi(0)$$

$$\vdots$$

$$\pi(k)=P^T\pi(k-1)=P^T\cdots P^T\pi(0)=\left(P^T\right)^k\pi(0)$$

Using this definition, we can define a stationary distribution π as a distribution with the property that $P^T\pi(k+1)=\pi(k)$, and according to *Ergodic Theorem*, under certain conditions, $\lim_{k\to\infty}\pi_k P^T=\pi_k$, for all starting distributions π.

## 3.4 Compute The PageRank Vector

The computation of PageRank vector exactly comes from the model of Markov Chain. Once we have Google Matrix, its adjusted PageRank computation is simply given by

$$\pi_{k+1}^T=\pi_k^T G\,. \qquad (3.4.1)$$

There are some variations on the computation of PageRank vector. It arises from a generalization of the random surfer idea. To compute the PageRank vector, we just need to find the stationary distribute of the Markov Chain. There are a number of methods out there to do so, but the feature of the Google matrix makes the power method a quite suitable one.

The power method is an iterative method that finds the vector $x_{k+1}$ from $x_k$ until the vector x converges to a desired value. The goal is to find dominant left-hand eigenvector of G corresponds to eigenvalue $\lambda = 1$. It is a simple solution to find the stationary distribution of Markov Chain. G can be expressed in terms of sparse matrix H as the equation below.

$$G = \alpha S + (1-\alpha)1/nee^T = \alpha(H + 1/nae^T) + (1-\alpha)1/nee^T = \alpha H + (\alpha a + (1-\alpha))1/ne^T .$$

The vector-matrix multiplication is executed on this H with about 10 non zeros in each row. In this way no matrix multiplication is need, and only vector matrix multiplication is used, which gives O(n) efforts to compute in each iteration. Furthermore, at each iteration, the power method only requires the storage of one vector, the current iterate. The irreducibility of the matrix P compliments the random surfing matrix E, guarantees the existence of the unique stationary distribution vector for the Markov equation. Convergence of the PageRank power method is governed by the primitivity of P. Because the iteration Matrix P is a stochastic matrix. If this stochastic matrix is not primitive, it may have several eigenvalues on the unit circle, causing convergence problems for the power method.

A primitive stochastic matrix has only one eigenvalue on the unit circle, all other eigenvalues have modulus strictly less than 1. This means that the power method applied to a primitive stochastic matrix P is guaranteed to converge to the unique dominant eigenvector the stationary vector $\pi^T$ for the Markov matrix and the PageRank vector for the Google matrix.

If we take the computation of PageRank vector in the manner that is close to power iteration. The iteration $x^{(k+1)} = \alpha P x^{(k)} + (1-\alpha)v$ where $x^{(0)} = v$ is equivalent to the power method. Let us see the error after a single iteration:

$x - x^{(k)} = [\alpha Px + (1-\alpha)v] - [Px^{(k)} + (1-\alpha)v] = \alpha P(x - x^{(k)})$. Let P, v be the matrix and vector for a PageRank problem. If $x^{(0)}$ = 1/n, then the error after k iterations will become $\|x - x^{(k)}\|_1 \le \|x - v\|_1 \alpha^k \le 2\alpha^k$. Set the values to be 0.99.

In the worst case, this method needs at most 3500 iterations to converge to a 1-norm error of $10^{-16}$. That gives a great performance of the power method.

## 3.5 Convergence of PageRank

Given the question of how the power method can have a speed convergence, the answer comes from the theory of Markov Chains. The asymptotic rate of convergence of the power method relies on the ratio of two eigenvalue. Let's say we have two eigenvalues from the matrix, denote $\lambda_1$ and $\lambda_2$. For stochastic matrix G, it is irreducible, which implies that $\lambda_1$=1, then $\lambda_2$ governs the rate of convergence, which is

$$\left| \lambda_2 / \lambda_1 \right|^k . \tag{3.5.1}$$

We can show that $\lambda_2 \le 1$. Numerically finding $\lambda_2$ is a hard task, but the good thing is that we do not have the manual find them one after another. If the spectrum of the stochastic matrix S is {1, $\lambda_1$, …, $\lambda_n$}, then the spectrum of Google Matrix G is {1, $\alpha\lambda_1$, …, $\alpha\lambda_n$}.

## 3.6 Mathematical Properties of PageRank

Let us jump out of the Markov Chain theory, and take a look at the general linear algebra, from which we can find some interesting conclusions regarding both computation and analysis of PagerRank. One claim of the linear system to be more efficient [Gleich 2014] will be illustrated here.

### 3.6.1 Linear System

This solution brought us out of the Markov theory and into the linear system. the original equation $\pi = G\pi$ can be written as $\pi(I - G) = 0$, so the goal is to find the left-hand null vector of I-G. We write the vector I-G in a stochastic and primitive way. With the normalized eigenvector problem, this can be rewritten as

$$\pi^T(I - \alpha S) = (1-\alpha)v^T . \tag{3.6.1}$$

We replace the matrix S with H, denote S=H+av$^T$, the linear system will become,

$$\pi^T(I - \alpha H - \alpha a v^T) = (1-\alpha)v^T \tag{3.6.2}$$

Here we can replace $\pi^T e = 1$, the computation of PageRank has become solving the linear system,

$$\pi^T(I - \alpha H) = v^T , \tag{3.6.3}$$

then with $\pi^T = x^T / x^T e$, it will produce the PageRank vector.

Let us take a look at some of the conclusions we can derive from the formula. We use the linear system because of the following reasons. In the linear system, the existence and uniqueness of the solution is immediate: the matrix I-G is a diagonally dominant M-matrix. The solution x is non-negative. There is only one possible normalization of the solution. We note that among the strategies to solve PageRank problems, those based on the linear system are both more straightforward and more effective than those based on the eigenvalue approach.

However, thinking of the storage issue, we have to conquer the large matrix problem. The matrix representing the whole web is huge, so there is no way to compute the linear system in a normal manner. Some parallel programming technique has to be considered and applied.

### 3.6.2 Sensitivity of PageRank

As we know, the equation of PageRank algorithm is expressed as a Google Matrix G. The outcome of computation of PageRank could vary based on a tiny change of its parameters. Now we are interested in how these parameters would affect the PageRank algorithm. There are three main aspects that contribute to the sensitivity of PageRank. The scaling parameter α, the hypermatrix and the personalized vector.

#### 3.6.2.1 Sensitivity with respect to α

It is interesting to study the effect of α. From the formula $G = \alpha S + (1-\alpha)e^T v$ , we know that α is the probability for the stochastic matrix that represents a user following the actual link on the web, and with different α value, we can see the computation with respect to α from 0 to 1 on the sample graph in Figure 3.1. What we see is that α affects the computation of PageRank. Low value of α brings us fast computation, but deviates from the original concept that user will follow the links on the page.

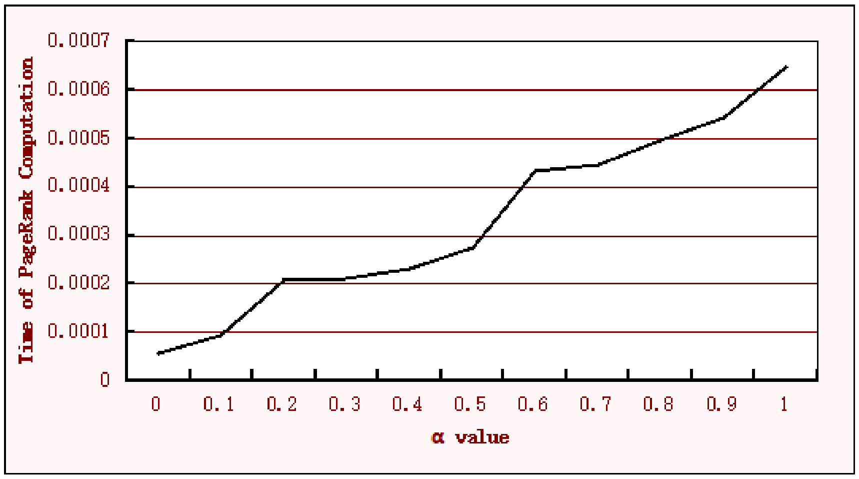


Figure 3.3 Time of PageRank Computation with respect to α from 0 to 1

In [Langville et. al. 2006], sensitivity with respect to α is given by the equation $\frac{\partial \pi^T(\alpha)}{\partial \alpha} = \frac{-v^T(I-S)}{(I-\alpha S)^2}$. It tells us how much the elements in PageRank vector vary when α changes. In order to do this, let 0 ≤ α+μ < 1, and set $G' = (\alpha+\mu)S + (1-(\alpha+\mu))eev^T$. Therefore, the error will become

$$\|\pi' - \pi\|_1 \le \frac{2}{1-\alpha}|\mu|. \tag{3.6.4}$$

The condition number bound 2/(1−α) is an increasing function in α, which shows that π is slightly more sensitive with larger α. This could be observed with some small perturbations of the graph structure S.

### 3.6.2.2 Sensitivity with respect to S

The sensitivity with respect to S can be found with the similar manner. The perturbed PageRank Matrix will give a different vector according to the power method, that is $\pi'^T G' = \pi'$, With the updated matrix $G' = \alpha(S+T) + (1-\alpha)eev^T$, we obtain the error $\pi'^T - \pi^T = \alpha\pi'^T E(I-\alpha S)^{-1}$. So the sensitivity depends on a condition number that is bounded by $\frac{\alpha}{1-\alpha}$.

$$\|\pi' - \pi\|_1 \le \frac{\alpha}{1-\alpha}\|E\|_\infty \tag{3.6.5}$$

A simple test is executed on the sample graph. We compared the result correspond to different web structure. The change comes from the addition link from D to B, and the structure is altered slightly. The fourth line of the stochastic matrix S is different.

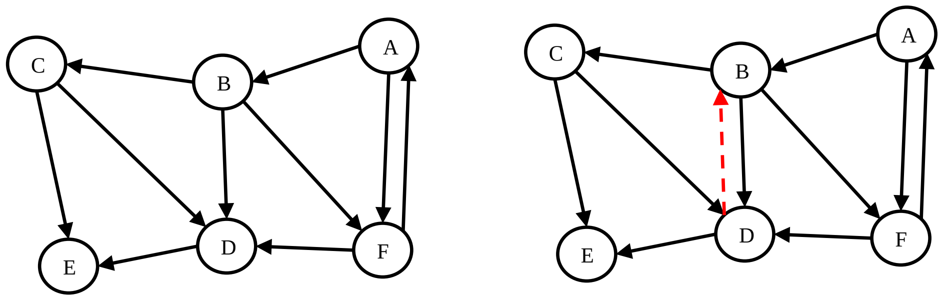


Figure 3.4 Adjusted sample web graph

$$\begin{bmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1/3 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/2 & 0 & 0 & 1/2 & 0 & 0 \end{bmatrix}$$

Then the result ranking of the new graph with an additional edge is provided in the table below,

Table 3.1 PageRank Score under Different Matrix

| α=0.85 | | α=0.1 | |
|---|---|---|---|
| S | S+T | S | S+T |

|  |  |  |  |
|---|---|---|---|
| 0.5646 | 0.5464 | 0.4242 | 0.4243 |
| 0.1135 | 0.3088 | 0.3359 | 0.3828 |
| 0.1485 | 0.1842 | 0.3598 | 0.3626 |
| 0.4070 | 0.4737 | 0.4412 | 0.4449 |
| 0.5591 | 0.3944 | 0.4681 | 0.4216 |
| 0.4099 | 0.4383 | 0.4049 | 0.4079 |

With the given graph, we showed that there are some changes in the PageRank vector under the same scaling factor. Despite this change, we have another observation that when α gets close to 1, the resulted vector is more sensitive. We can have a conclusion. For small α, PageRank is insensitive to the variation, but as α becomes larger, PageRank will become more sensitive to small perturbations. Once α gets close to 1, PageRank is very sensitive to small changes.

**3.6.2.3 Sensitivity with respect to v**

We start with PageRank sensitivity with respect to personalization vector v. Let v+t be the updated personalization vector, and $G^{'} = \alpha S + (1-\alpha)ee(v+t)^T$. The perturbed PageRank vector is $\pi^{'}$, where. The error bound for $\pi^{'}$ contains a condition number that is bounded by one

$$\| \pi^{'} - \pi \|_1 \le \| t \|_1 \qquad (3.6.6)$$

A simple test is executed on the sample web graph. Set v to be $G = \alpha H + 2(\alpha a + (1-\alpha))/ne^T$, and let α be 0.85 and 0.1 respectively. We can acquire the output as follow,

Table 3.2 PageRank Score under Different Personalized Vector

| $v = \frac{1}{n} e^T$ | $v = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \end{bmatrix}$ |
|---|---|
| 0.2561 | 0.3360 |
| 0.2159 | 0.3259 |
| 0.1390 | 0.1382 |
| 0.4501 | 0.4498 |
| 0.7529 | 0.6600 |
| 0.3146 | 0.3519 |

According to [Langville et. al. 2006], as α approaches to 1, the PageRank becomes increasingly sensitive. What we can conclude here is that it does not have significant difference as α changes. Some of the values are flipped, but it is hard to determine from the test whether it is sensitive.

## 3.7 Personalized PageRank

We have seen the fundamental concept behind PageRank and its mathematical background. We know that this algorithm is executed offline, which means it will compute the rank and update global score for the web graph. During query time, the search engine will bring back you results that is relevant to the query term. What we are trying to see here is a little variation about PageRank. The change is made on teleportation matrix E. Recall that in the PageRank model that Brin and Page suggested, the teleportation matrix is $E=1/nee^T$, meaning the user is equally like to jump from one page the another. However, this matrix only gives a uniform distribution and does not represent user's personal reference. Rather than use $E = \frac{1}{n} ee^T$, ev$^T$ is applied to be the personalized matrix. This seems to be the origin intent in introducing personalized PageRank. Some researchers have created pseudo-personalized PageRank. [Haveliwala et. al. 2003] adapted the idea and created topic-sensitive PageRank. In personalized PageRank, this modification of matrix ev$^T$ also retains most of the properties and advantages of power method. When the PageRank formula becomes $G = \alpha S + (1-\alpha) ev^T$, then the power method will become,

$$\pi^{(k+1)T} = \alpha \pi^{(k)T} H + (\alpha \pi^{(k)T} a + 1 - \alpha) v^T. \qquad (3.7.1)$$

$\pi^T (v^T) = v^T$ is a function of v$^T$, meaning different personalized vector v produce different PageRanking. We know that no matter what the initial distribution of $\pi$ is, the PageRank vector will converge to a unique distribution depending on the stochastic matrix, so the ranking vector mainly rely on the initial value of v. It can also be deemed localized PageRank vector. When we solve a localized PageRank problem in a large graph, and the nodes we select for teleportation lie in a region that is somehow isolated, yet connected to the rest of the graph. Then the final PageRank vector is large only in this isolated region and has small values on the remainder of the graph because once the there is a small change of the weight on some nodes, all they can affect are the nodes close to them in that region [Gleich 2014]. This behavior is exactly what most use of localized PageRank want: they want to find out what is nearby the selected nodes and far from the rest of the graph.

However, there is one thing that need to be aware of, or even put effort to, is the computation of personalized PageRank during the query time. Due to user's preference, the personalized score cannot be computed beforehand, there has to be a query processing period and extra computation time and that is a difficult problem to handle.

## 3.8 Performance Evaluation

For PageRank algorithm, since it is query-independent, the global score will not able to deliver too much information. So we would like to see how it performs on evolving graph. This could somewhat tell us what is going to happen to the rank of Pages in the real world when the web graph changes.

We use the web crawler to retrieve 20,000 and 1,084,390 edges. Starting with 10000 nodes subgraph, we ran PageRank ten times on this web graph with each time 1000 nodes added to the graph. Then, we have the results retrieved in the table below, together with the degree distribution for 20000 nodes.

Table 3.3 Top 15 results retrieved by pagerank algorithm on the *OU* web graph

| |
|---|
| 1 http://www.ou.edu/ |
| 3 http://www.ou.edu/web.html |
| 36 http://www.ouhsc.edu/ |
| 49 http://www.facebook.com/uoklahoma |
| 50 https://twitter.com/uofoklahoma |
| 51 http://www.ou.edu/content/publicaffairs/WebPolicies/copyright.html |
| 52 http://www.ou.edu/content/web/resources_offices.html |

| |
|---|
| 53 http://www.youtube.com/universityofoklahoma |
| 54 http://www.ou.edu/web/socialmediadirectory.html |
| 55 http://www.ouhsc.edu/ |
| 56 http://www.ou.edu/content/web/resources_offices.html |
| 57 http://ouhsc.edu/hipaa/ |
| 58 http://www.ou.edu/content/web/landing/policy.html |
| 105 http://www.ou.edu/ousearch.html |
| 106 http://hr.ou.edu/jobs/ |

This is the result for the graph with 20000 nodes. With the slight change of the graph, the top 15 results always come from this page set.

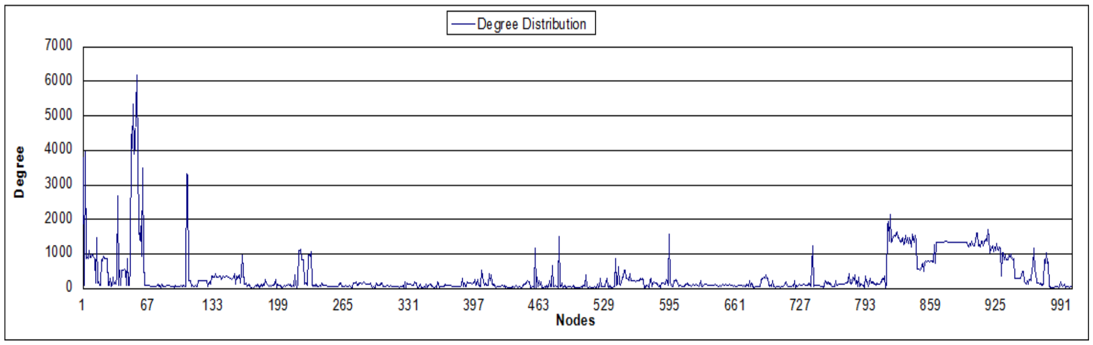
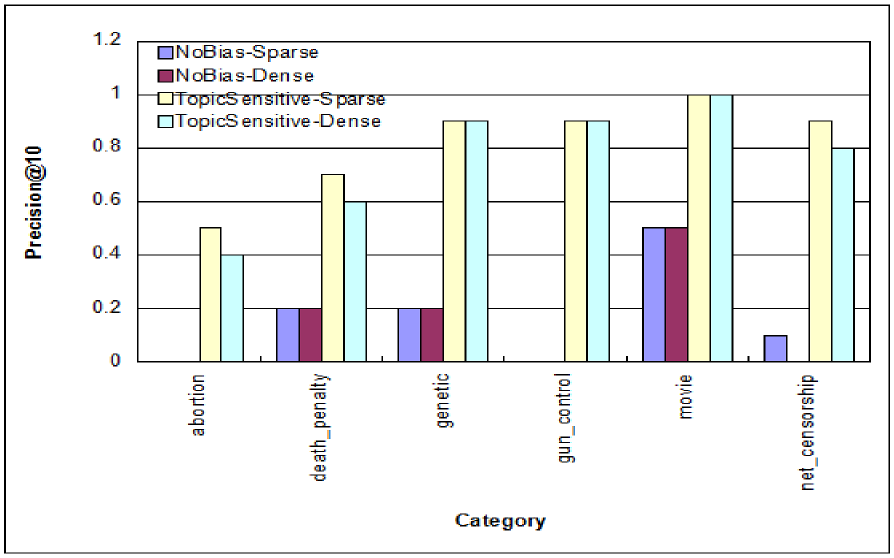


Figure 3.5 Degree Distribution of OU web graph with 20000 nodes. With the graph size going from 10000 to 20000. The degree distribution roughly follows the same pattern. Nodes in Table 3.3 always have higher degree.

With the result we acquired from the algorithm, we found that despite different number of web pages, the top score pages always remain the same, though the rank could slightly change. With more pages added to the web graph, if the degrees of popular pages are still higher than the others, they will are more likely to get a higher score. The change of the big web graph does not affect too much to the final score as long as the “important” page keeps being pointed by more other pages. Meanwhile, these important pages will be always shown as the top search result. One reason for this that we can find is that those pages are all in the core components of the graph. This in accordance with the concept of PageRank that important pages are pointed by and pointing to many pages. In a large web graph, an addition or removal of less important nodes do not harm the search result. However, if someone maliciously add a lot of links to some pages, that could change the entire ranking vector.

Once we adapted personalized PageRank, we would like to see how it affects the ranking of pages with corresponding topics. To construct the sample graph, we used graphs from http://www.cs.toronto.edu/~tsap/experiments/datasets/index.html. We have two slightly different graphs to test on. The first one is original graph with no biased personalized vector. In the Six categories including abortion, death_penalty, genetic, gun, movies and net_censorship are selected. These graphs are put together as a diagonal matrix. In order to connect those categories, we choose two each time and randomly so that there will be edges going from one subgraph to another. In the sparse graph, we selected 50 nodes from each category. The web graph contains 16,799 nodes and 64,813 edges. In the little bit more dense one, the number of edges added between two categories is 500, and that makes 69313 edges. We computed the global score of the graph with and without personalized PageRank. The measurement we use here is mean average precision. We skipped the pre-processing of query classifying step in topic sensitive PageRank.

In this experiment, we simply change the personalized vector, and bias it toward the category we are interested in. That is not quite different from the query-dependent PageRank vector introduced in [Haveliwala 2003]. The query-dependent PageRank vector is a convex combination of biased vectors. That is, $\pi^T = \beta_1 \pi^T (v_1^T) + \beta_2 \pi^T (v_2^T) + \cdots + \beta_n \pi^T (v_n^T)$. In this experiment, as an example, a query on *abortion* falls in category Abortion, so PageRank vector associated with that topic will be given more weights.



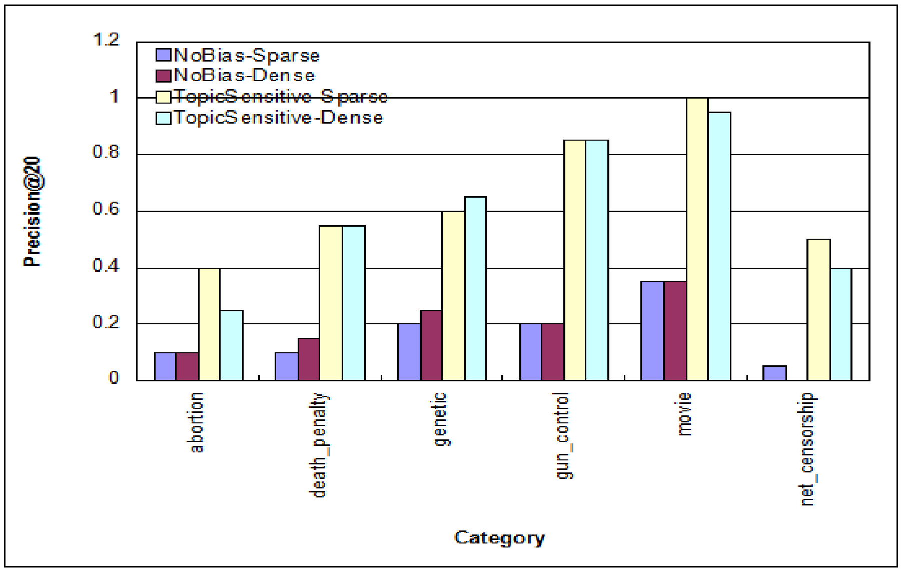


Figure 3.6 a) Precision@10 of each category for test queries b) Precision@20 of each category for test queries

The precision of the two ranking techniques for each test query is shown in Figure 3.4. The average precision for the rankings induced by the personalized pagerank is substantially higher that unbiased PageRank score. These results suggested the effectiveness of query-dependent pagerank. Thus there are essetially some improvements in the topic sensitive pagerank scheme.

## 3.9 Conclusion

As a primary algorithm used by Google, PageRank has shown a good performance in real world uses.  PageRank is relatively suitable for search engine that requires a universal result base the term. Users may want any type of documents related to the search terms to be presented we they search, PageRank is able to bring all possible feedback to them.

In terms of computational performance, it only takes tens of steps of iteration and a few seconds to retrieve the result according to the query. Small perturbation of the gigantic web graph will not cause significant changes to the search result.

Currently, PageRank update their ranking scores about ones a month, and that is also a general ranking of all pages in the web. If there are some technique that could extract richer documents, it could make a difference for the search result. However, there is not principle for a “good” search result. The performance of search engine is largely depend on human judge. How to optimize search engine and serve different search results based on users search manner would be an interesting topic.

# Chapter 4

# HITS Algorithm

## 4.1 HITS Introduction

PageRank algorithm performs a query-independent method to compute the web pages ranking vector, which gives the ranks of pages before executing the query. On the contrary, HITS algorithm showed a different way of link analysis. In this work, HITS focuses on the use of links for analyzing the collection of pages relevant to a broad search topic, and for discovering the most “authoritative” pages on such topics.

This work originates in the problem of searching on the web, which we could define roughly as the process of discovering pages that are relevant to a given query. Two types of obstacles are mainly what this algorithm focus on. If there are very few pages that contain the required information, it is often difficult to determine their identities; when one expects to find many relevant pages on the web, the difficulty lies in that the number of pages that could reasonably be returned is far too large. One other obstacles the algorithm is addressing is the issue of accurately modeling authority in the context of a particular query topic. Links afford the opportunity to find some potential authorities through the pages that point to them, and this offers a way to bring more prominent pages that are potentially meaningful.

In this thesis, we want to describe this link-based model for finding the authority, and show how it leads to a method that can identifies relevant, authoritative web pages for search topics. We also give a mathematical analysis on

HITS algorithm and its performance.

## 4.2 The Original HITS Algorithm

The algorithm operates on focused subgraphs of a web graph that is retrieved from a web crawler. The technique for constructing such subgraphs is designed to produce small collections of pages likely to contain the most authoritative pages for a given topic. Suppose that we have a query t. The highest ranked pages $R_t$ are firstly collected as a root set by a text based search engine. Then we can use the root set to produce a set of pages Sq that is considered as a proper page collection for the algorithm. For a strong authority for the query, if it is not in the set $R_t$, it is quite likely to be pointed to by at least one page in $R_t$. To add pages to the base set, the follow strategy is applied.

For each page p in $R_t$, from all pages that is pointing to p, we randomly choose 50 pages and put into the set. Then we also add to the base set all the pages that is pointed by pages in the root set. Hence, The page set is increased by expanding $R_t$ along the links that enter and leave it.

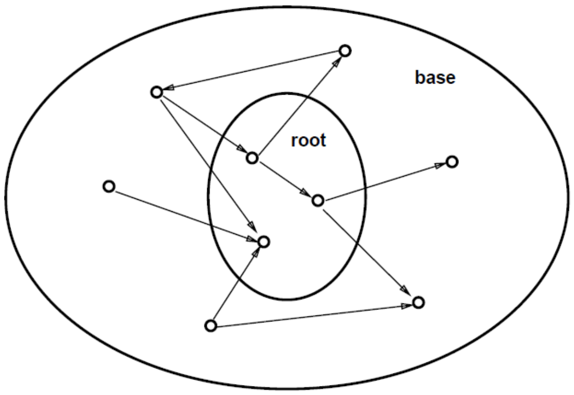


Figure 4.1 Subgraph containing root set and base set

Next, we hope to extract the authorities from the overall collection of pages through an analysis of the link structure of $G_t$. To deal with the problem, the web graph will be divided into two categories, hubs and authorities. An authority is a page with incoming links and a hub is a page with outgoing links. Hubs and authorities exhibit what could be called a mutually reinforcing relationship: A good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs. Every page is measured both as hub and authority and keeps both score, which expresses the mutual reinforcement relationship between hubs and authorities.

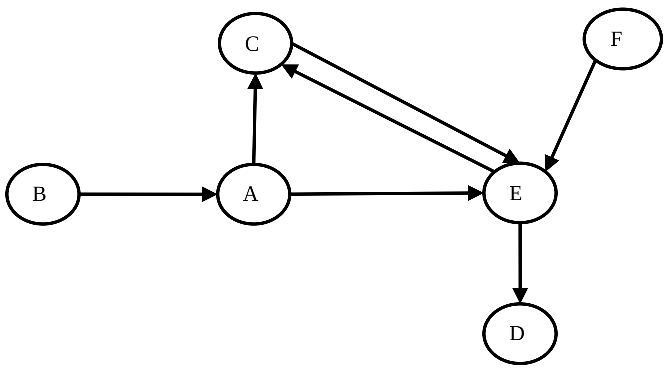


Figure 4.2 An Example of Directed Neighborhood Graph

For each query, the terms will filter in the pages containing these terms. Then, the graph will be expanded by adding pages that point to the subset. This process will generate a relatively huge graph.

## 4.3 The Computation of HITS

To generate the ranking vector of the pages with HITS, we first select a set of pages to build a neighborhood graph. With a query, we select a smaller set of pages S which contain most of the authorities. In this set, there are some pages owning the higher rank, we refer them as the root set R, and the produce set S. An authority page may not in the root set, but could be pointed by a page in the root set, and the pages that point to the authorities are hubs.

As is mentioned above, every page has both authority score and hub score. Let $e_{ij}$ be the directed edge from node i to j. Each page initially is assigned an authority score and a hub score. Then HITS refines these scores successively by iteration.

$$x_i^k = \sum_{j:e_{ji} \in E} y_j^{k-1} \tag{4.3.1}$$

$$y_i^k = \sum_{j:e_{ji} \in E} x_j^{k-1} \tag{4.3.2}$$

Replacing these equations with matrix representation, we derive the equation of matrix-vector multiplication.

$$x^k = L^T y^{k-1} \tag{4.3.3}$$

$$y^k = Lx^k \tag{4.3.4}$$

This can derive a simplified substitution.

$$x^k = L^T L x^{k-1} \tag{4.3.5}$$

$$y^k = LL^T y^{k-1} \tag{4.3.6}$$

Follow these equations we have presented, the computation process of HITS algorithm will be quite straightforward. Given the web graph, each node will contain two properties: the hub score and the authority score .To implement HITS, a graph of query terms is built. Let us suppose the graph of Figure 4.2 is extracted according to the text based search engine. The corresponding adjacency matrix is formed as,

$$L = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

We follow the iteration process as stated previously and performed the computation of the HITS algorithm on the sample graph in Figure 4.2. The respective hub matrix and authority matrix and their scores are:

$$L^T L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad LL^T = \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Table 4.1 Authority Score and Rank

| | | | | | | |
|---|---|---|---|---|---|---|
| Authority score | 0 | 0.1667 | 0.3333 | 0.1667 | 0.3333 | 0 |
| Authority rank | 6 | 4 | 2 | 3 | 1 | 5 |

Table 4.2 Hub Score and Rank

| | | | | | | |
|---|---|---|---|---|---|---|
| Hub score | 0 | 0 | 0.3660 | 0.3660 | 0.2679 | 0 |
| Hub rank | 6 | 5 | 2 | 1 | 3 | 4 |

## 4.4 HITS Stability

The stability problem of HITS algorithm can be examined by the formula according to [Langville et. al. 2006],

$$\sin(x, x') \le \frac{\|E\|_2}{\lambda_1 - \lambda_2} \tag{4.4.1}$$

It tells us that the separation between the two dominant eigenvalues governs the sensitivity of the HITS vector. If the eigengap is too large, then the authority vector is insensitive to small changes. On the other hand, the vector may be sensitive if the eigengap is small. This is also shown in [Andrew et. al. 2006] that the stability of this eigenvector under small perturbations is determined by the eigengap of S, where S=L$^T$L.

Let S □be given and $\lambda_1$ be the principal eigenvectors and $\delta$ be the eigengap of S. Assume the maximum out-degree of every web page is bounded by d. For any ε > 0, suppose we perturb the web graph by adding or deleting at most k links from one page, where $k < (\sqrt{d+\alpha} - \sqrt{d})^2$ where $\alpha = \varepsilon\delta / (4 + \sqrt{2}\varepsilon)$. Then the perturbed principal eigenvector $\lambda_2$ of the perturbed matrix S' satisfies,

$$\|\lambda_1 - \lambda_2\|_2 \le \varepsilon \, . \tag{4.4.2}$$

The proof of this theorem is in the paper. To give a detailed analysis of stability, we can have an intuition from the following perspective and simply think about the stability of HITS with regard to the small change. We will focus on the authority side of the rank. The first example we showed is a 10 nodes' graph with 18 edges.

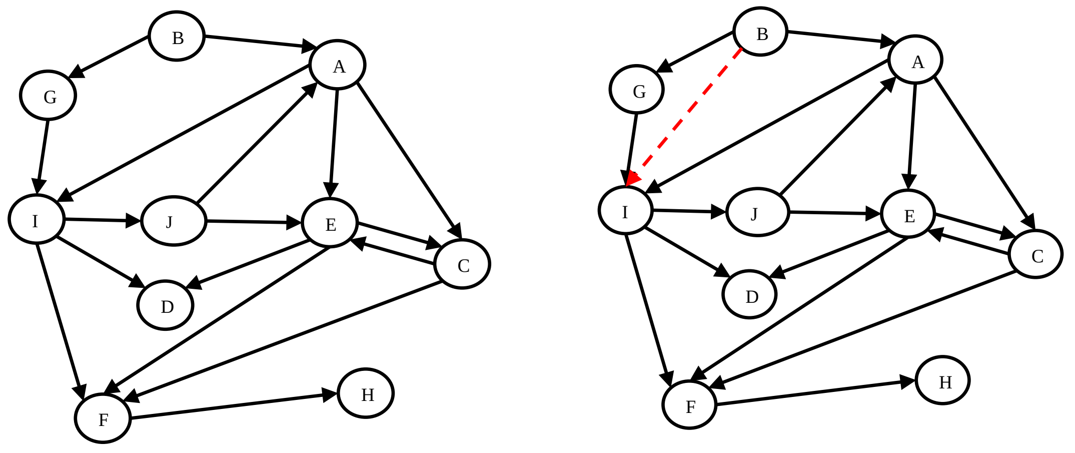


Figure 4.3 A Neighborhood Graph with slight change

Table 4.3 Authority Score of the graph regarding the slight change

| Original Graph | | Updated Graph | |
|---|---|---|---|
| Authority Score | Rank | Authority Score | Rank |
| 0.0538 | 7 | 0.0863 | 6 |
| 0 | 10 | 0 | 9 |
| 0.1609 | 3 | 0.1466 | 3 |
| 0.1589 | 4 | 0.1270 | 5 |
| 0.2252 | 2 | 0.2173 | 1 |
| 0 | 9 | 0 | 9 |
| 0.0100 | 8 | 0.0401 | 8 |
| 0.2304 | 1 | 0.1888 | 2 |
| 0.0883 | 5 | 0.1372 | 4 |
| 0.0725 | 6 | 0.0567 | 7 |

The first example we showed is a 10 nodes' graph with 18 edges. We computed the authority scores and then made a slight change, which is shown in Figure 4.3. Conceptually, there should be only one change on the authority side, and that should be node I because it received one more incoming link. However, we find that authority score of nodes A, C, D, E, G, H, I, J are all different, and their rank are completely changed accordingly while there is only one link added to the graph.

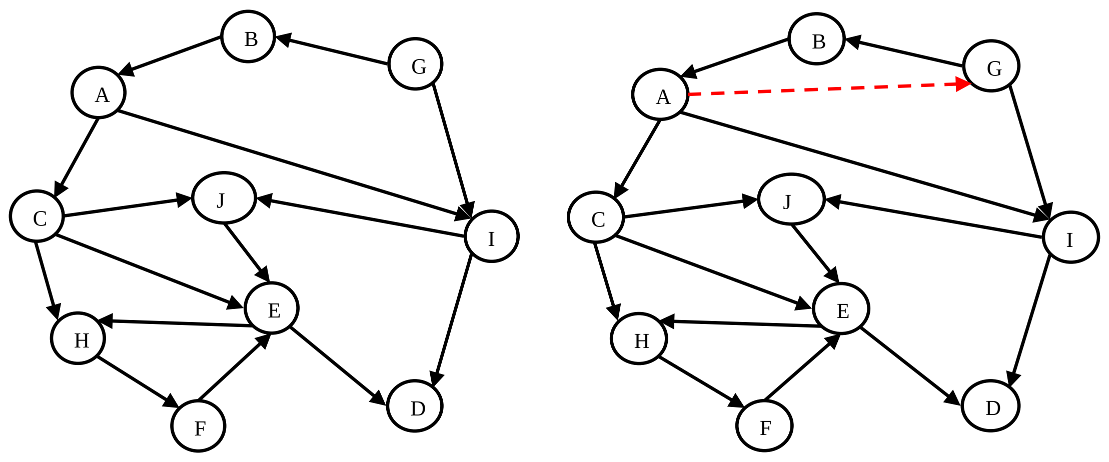


Figure 4.4 A less Sparse Neighborhood Graph from Figure 4.3 with slight change

Now we have made the graph less dense. We also compute the authority score and then made a slight change, which is shown in Figure 4.4.

Table 4.4 Authority Score of the graph regarding the slight change

| Original Graph | | Updated Graph | |
|---|---|---|---|
| Authority Score | Rank | Authority Score | Rank |
| 0 | 8 | 0 | 9 |
| 0.0424 | 7 | 0.0424 | 7 |
| 0.1649 | 3 | 0.1649 | 3 |
| 0.1639 | 4 | 0.1639 | 4 |

|  |  |  |  |
|---|---|---|---|
| 0.227 | 2 | 0.227 | 2 |
| 0 | 8 | 0 | 9 |
| 0 | 8 | 0.02 | 8 |
| 0.237 | 1 | 0.237 | 1 |
| 0.09 | 5 | 0.09 | 5 |
| 0.0749 | 6 | 0.0749 | 6 |

In this example, there is also only one edge added to the graph. As a result, we find that the authority score almost has no change. Only node G has a score from 0 to 0.02 because it has an incoming link. In the two cases above, the first sample graph has an eigengap of 0.274 and the second one has an eigengap of 1. This gives an intuition that the larger eigengap will result in a better stability.

In general, the sensitivity of HITS follows this rule. We can think about some special cases in real world web graph that this could happen in a local community, and the computation need to be re-evaluated to prevent such irregularity from happening.

## 4.5 HITS Convergence

Since the actual magnitude of the hub and authority values tend to grow with each update, there is a direct picture of what's happening in the iteration of HITS computation: the authority and hub vectors are the results of multiplying an initial vector by larger powers of $L^TL$ and $LL^T$ respectively. This is little bit more complicated than PageRank convergence.

Here is the theory behind this. Any symmetric matrix M with n rows and n columns has a set of n eigenvectors that are all unit vectors and all mutually orthogonal, and they form a basis for the space $R_n$. Here, both $LL^T$ and $L^TL$ are symmetric, so the proof will follow this way. Let $z_1, z_2, \dots, z_n$ be the set of eigenvectors with corresponding eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ respectively. If we order the eigenvalues of $LL^T$ to be $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, and suppose that $|\lambda_1| > |\lambda_2|$, the rate of convergence is given by

$$[\lambda_2(L^TL)/\lambda_1(L^TL)]^k . \tag{4.5.1}$$

The complete proof is given in Appendix.

## 4.6 Performance Evaluation

In order to see the performance of HITS algorithm, we conducted two experiments on relatively large scale web graphs. The first graph we used is the computational_geometry on the website http://www.cs.toronto.edu/~tsap/experiments. This graph contains 1,226 nodes and 3,953 edges. The result is shown in the table below.

Table 4.5 Top 10 authority results from computation geometry category with HITS

| |
|---|
| [O] http://www.ics.uci.edu/~eppstein/geom.html<br><br>Geometry in Action |
| [R] http://www.geom.umn.edu/software/cglist<br><br>Directory of Computational Geometry Software |
| [R] http://www.cs.uu.nl/CGAL |

| |
|---|
| The former CGAL home page |
| [O] http://www.ics.uci.edu/~eppstein/junkyard<br><br>The Geometry Junkyard |
| [R] http://www.scs.carleton.ca/~csgs/resources/cg.html<br><br>Computational Geometry Resources |
| [O] http://www.geom.umn.edu<br><br>The Geometry Center Welcome Page |
| [O] http://www.ics.uci.edu/~eppstein<br><br>David Eppstein |
| [O] http://www.mpi-sb.mpg.de/LEDA/leda.html<br><br>LEDA - Main Page of LEDA Research |
| [O] http://www.cs.sunysb.edu/~algorith<br><br>The Stony Brook Algorithm Repository |
| [O] http://www.inria.fr/prisme/personnel/bronnimann/cgt<br><br>CG Tribune |

Table 4.6 Top 10 hub results from computation geometry category with HITS

| |
|---|
| [O] http://www.ics.uci.edu/~eppstein/geom.html<br><br>Geometry in Action |
| [O] http://www.ics.uci.edu/~eppstein/junkyard<br><br>The Geometry Junkyard |
| [O] http://www.uci.edu |

| |
|---|
| University of California, Irvine |
| [R] http://www.geom.umn.edu/software/cglist<br><br>Computational Geometry Software |
| [R] http://www.cs.uu.nl/CGAL<br><br>The former CGAL home page |
| [R] http://www.scs.carleton.ca/~csgs/resources/cg.html<br><br>Computational Geometry Resources |
| [O] http://www.ics.uci.edu/~eppstein<br><br>David Eppstein |
| [O] http://www.mpi-sb.mpg.de/LEDA/leda.html<br><br>LEDA - Main Page of LEDA Research |
| [O] http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html<br><br>Mesh Generation & Grid Generation on the Web |
| [O] http://www.geom.umn.edu<br><br>The Geometry Center Welcome Page |

It is not hard to see that most of the high rank pages are in root set or pointed by pages in the root set. One common thing it has is that their all have more incoming links. However, because of the computational property of HITS, the top hub pages are more likely the same as authority pages and it does not offer to much option for users, although most of the user only care about no more than 10 results.

Our next test is based on a larger web graph. The web crawler program crawled

50,000 nodes with 2,721,092 edges with breadth first search. We used the dataset on the matlab program to simulate the computation of HITS algorithm. Since HITS is a query dependent algorithm, we specified the query *University of Oklahoma* and extracted the neighborhood graph from the original web graph. If we want to include every single page that contains the query, it could be over 1000 pages and that is inappropriate for the algorithm. So we extracted 450 pages that is relatively rich in query text and view them as our root set. Then we add those pages that pointing to the root set. The size of the neighborhood graph and the number of back links pointing to authority pages could be those factors that affect the ranking. We also use normalized discounted cumulative gain as our measurement for HITS.

The normalized discounted cumulative gains (NDCG) measure [Järvelin, Kalervo et. al. 2002] discounts the contribution of a document to the overall score as the document's rank increases. Such a measure is now considered an appropriate measurement for search engines and widely used by the studies of information retrieval. NDCG values are normalized with the higher value being a "perfect" ranking scheme that completely agrees with the assessment of the human judges. The discounted cumulative gain at a particular rank-threshold k (DCG@k) is defined to be

$$P_{T_k} = \sum_{k=1}^{T_k} \frac{rel(k)}{\log(i+k)}, \tag{4.6.1}$$

where rel(j) is the rating (0=detrimental, 1=bad, 2=fair, 3=good, 4=excellent, and 5=definitive) at rank j. The NDCG is computed by dividing the DCG of a ranking by the highest possible DCG that can be obtained for that query. Finally, the NDGC of all queries in the query set are averaged to produce a mean NDCG.

One thing to be mentioned is that if we simply construct the web graph according to the number of backlinks and include all pages that pointing to the root set, with the query term of College of Engineering, the top ranking pages we will retrieve could entirely harm the search result as shown following.

Table 4.7 Searching result without separated domain name

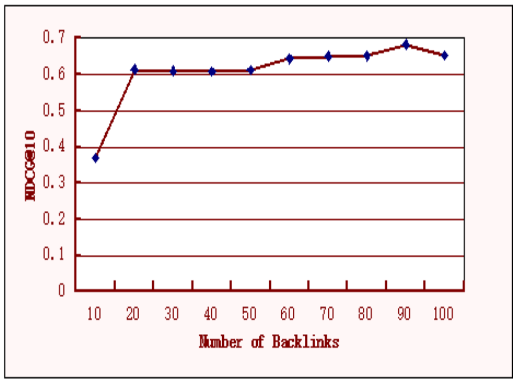
| |
|---|
| http://www.ou.edu/web.html |
| http://www.ou.edu/content/web/resources_offices.html |
| http://www.ou.edu/#twitter |
| http://www.ou.edu/#facebook |
| http://www.ou.edu/sustainability.html |
| http://hr.ou.edu/jobs/ |
| http://www.ou.edu/content/web/landing/policy.html |
| http://www.ou.edu/content/publicaffairs/WebPolicies/copyright.html |
| http://www.ou.edu/content/web/resources_offices.html |
| http://www.ou.edu/content/web/socialmediadirectory.html |

To overcome this problem, one method we tested here can be described as follows. We divide the web pages into two groups with respect to their domains. We extract the root set according to their domain name. Then once we add pages that pointing to the root set pages to the base set, they should not be under the same domain. Once we run the program, we see a difference.

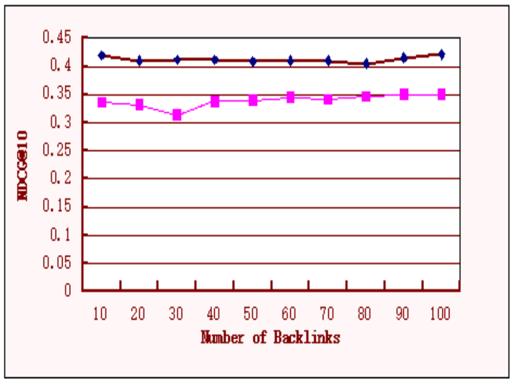
Table 4.8 Searching result with separated domain name

| |
|---|
| http://www.ou.edu/content/web/socialmediadirectory.html |
| http://www.ou.edu/web/socialmediadirectory.html |
| http://www.ou.edu/content/admissions/academics.html |
| http://www.ou.edu/content/publicaffairs/archives/OUReceives1MillionGrantfromtheMellonFoundationforContinuedDevelopmentofaDigitalLatinLibrary1.html |
| http://www.ou.edu/content/publicaffairs/archives/OUProfessorDevelopingVaccinetoProtectGlobalCommunitiesfromMalaria.html |
| http://www.ou.edu/content/visit/visitor_center_merchandise.html |
| http://www.ou.edu/content/give/where-to-give.html |
| http://www.ou.edu/content/coe.html |
| http://www.ou.edu/content/admissions/academics.html |
| http://www.ou.edu/content/enrollment/final_exams.html |

From the Tables above, we find that pages under the same domain could be a little navigational, most of the pages will point to a popular page regardless of what our query term is, so it takes away the score. Thus, no matter how you adjust the graph, SALSA does not offer too much help. However, with the adjustment that the pages under different domains are separated, the result seems to be much better.



(a)



(b)

Figure 4.5 a) NDCG of HITS in terms of query University of Oklahoma b) NDCG of HITS in terms of query College of Engineering with and without domain separation

The result in Figure 4.6 does not necessary mean whether HITS algorithm can bring all perfect result, Figure 4.6 a) just shows that when the neighborhood graph gets larger, the relevance score is getting slightly higher. The reason could be that give a neighborhood graph, in-degree is still the dominant factor for the ranking. In the neighborhood graph with fewer nodes, the portion of the nodes that are pointing to the top ranking pages is slightly larger than bigger ones. But in Figure 4.6 b), we cannot clearly see such phenomenon. The reason we suspect is that the ranking score might be related to construction of the subgraph. If the number of important nodes keeps the same in the graph, the rank will not change significantly. But we can see the improvement with the separation of domain. The only thing that could harm the ranking is number of pages that should be included in the set. No matter how we divide the group of pages, there might be some good pages not being included in the subgraph or not receiving a high score during the computation. These could be a concern of the next step of HITS algorithm.

## 4.7 Conclusion

HITS algorithm performs in a different manner from PageRank. It constructs a page set that is highly relevant to the search term and compute the score for only the pages in that set, and brings the notion of hub and authority, which gives user an option of browsing if the high rank pages are not satisfying.

An apparent advantage of HITS algorithm is the size problem. The size of the matrices are small so the computation will always be quick enough, although from the mathematical perspective it could be slow. However, as a query-dependent algorithm, HITS has to construct a neighborhood graph and this has to be done for each query. The topic drift problem is another big issue. In order to bring the proper ranking score, the process of the neighborhood graph before computation has to be strict so as not to bring unexpected pages to harm the rank.

HITS algorithm is not a very good choice for universal query as it only has limited number of options. This will be a nice choice for a local community where the there a few pages that is obviously more relevant than others.

# Chapter 5

# SALSA Algorithm

## 5.1 SALSA Introduction

While preserving the theme that Web pages pertaining to a given topic should be split to hubs and authorities, we here replaced Kleinberg's Mutual Reinforcement approach by a new stochastic approach so that the coupling between hubs and authorities is less tight.

The intuition behind this approach is the following: consider a bipartite graph G, whose two parts correspond to hubs and authorities, where an edge between hub H and authority A means that there is a link from H to A. Then, authorities and hubs pertaining to the same topic of the pages in G should be highly relevant to the query. Thus, we will attempt to identify these sites by examining certain random walks in G, under the condition that such random walks will tend to visit these highly visible sites more frequently than other, less connected sites. We show that in finding the principal communities of hubs and authorities, both Kleinberg's mutual reinforcement approach and the stochastic approach employ the same meta-algorithm on different representations of the web graph.

In this thesis, a mathematical analysis about how SALSA performs on different type of graphs will be given. We will compare the results of applying SALSA to the results derived by Kleinberg's approach, as SALSA is a modification based upon HITS. In certain scenarios, the Tightly Knit Community (TKC) effect hampers the ability of the mutual reinforcement approach to identify meaningful authorities.

SALSA can sometimes find meaningful authorities in collections where the mutual reinforcement approach fails to do so. We want to demonstrate whether SALSA is less vulnerable to the TKC effect.

## 5.2 SALSA Algorithm

In SALSA, link analysis is handled with the idea of ranking Web sites using random walks. Random walk is an old but classic concept applied by Google. It incorporates stochastic information into its ranking of web pages. The PageRank algorithm examines a single random walk on the entire Web. Hence, the ranking of Web sites in Google is query independent, and no distinction is made between hubs and authorities. SALSA integrated the property of PageRank on HITS, showing a different type of random walk.

### 5.2.1 SALSA Structure

To perform SALSA algorithm, we need to build a bipartite undirected graph G'. There are some collections of its structure included in this graph:

$$V_h = \{ S_h \mid S \in C \text{ and out-degree}(S) > 0\}$$

$$V_a = \{ D_a \mid D \in C \text{ and in-degree}(D) > 0\}$$

$$E = \{ (S_h, D_a ) \mid S \rightarrow D \text{ in } C\}.$$

where $V_h$ is the hub side of G', and $V_a$ is the authority side of G'. Each page $S \in C$ is represented in G' by one or both of the nodes $S_h$ and $D_a$. Each link $S \rightarrow D$ is represented by an undirected edge connecting $S_h$ and $D_a$.

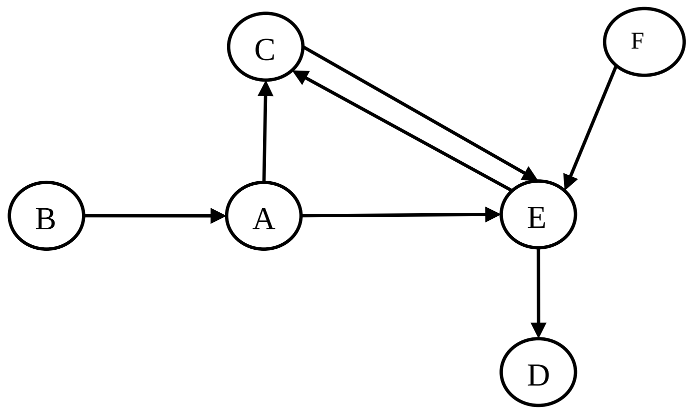


Figure 5.1 Neighbor Graph of 1 and 6

Here we give the partition of the graph from Figure 2. On this bipartite graph we will perform two distinct random walks. Each walk will only visit nodes from one of the two sides of the graph, by traversing paths consisting of two edges in each step. Each walk visits one side of the graph, and the two walks will naturally start off from different sides of G'.

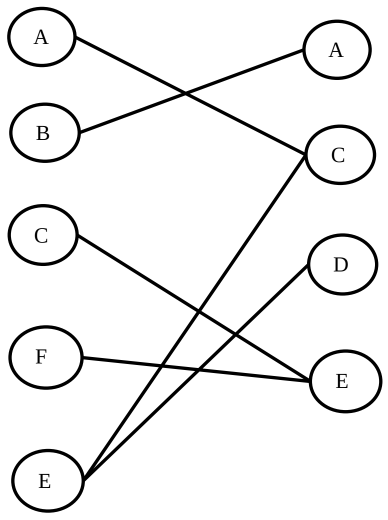


Figure 5.2 Bipartite graph of a neighbor graph

### 5.2.2 Random Walk on Bipartite Graph

Unlike in a “conventional" random walk, each step in the SALSA algorithm always traverses two links, one forward and one backward. That is, starting on the left-hand side of the bipartite graph, the random walk will always follow one of the links back to the left-hand side; similarly, if starting on the right-hand side, the random walk will always end up back on the right-hand side. This approach also relies on the notion of Markov Chain. Authoritative pages on a topic should be visible from a lot of pages in the graph. So the random walk on this graph should visit these authorities with high probability. PageRank examines a single random walk on the entire WWW, and no distinction is made between hubs and authorities. In SALSA, the random walk is generated by traversing two consecutive links, thus analyzing both chains and give each page two distinct scores, a hub score and an authority score. In this way, there will be a transition between two nodes if there is a path connecting them. This gives us a general equation of the transition probability,

$$P(i,j) = \sum_{k:i\to k,k\to j} \frac{1}{in(i)} \cdot \frac{1}{out(k)} \qquad (5.2.1)$$

where p(i, j) is the probability to go from i to j through node k, in(i) is the indegree of i and out(k) is the out degree of k. Analyzing these chains separately naturally distinguishes between the two aspects of each page. We now define two stochastic matrices:

The hub matrix H, defined as follows:

$$\mathrm{h}_{i,j} = \sum_{\{k|(i_h,k_a),(j_h,k_a)\in G'\}} \frac{1}{d(i_h)} * \frac{1}{d(k_a)} \qquad (5.2.2)$$

The authority-matrix A, defined as follows:

$$\mathrm{a}_{i,j} = \sum_{\{k|(k_h,i_a),(k_h,j_a)\in G'\}} \frac{1}{d(i_a)} * \frac{1}{d(k_h)} \qquad (5.2.3)$$

A positive transition probability $a_{i,\,j} > 0$ implies that a certain page k points to both pages i and j, and hence page j is reachable from page i by two steps: retracting along the link $k \to i$ and then following the link $k \to j$.

## 5.3 SALSA Computation

With the definition above, we can derive the matrix we need in the following way. Let L to be the adjacency matrix of the graph. Then let Lr be the matrix which results by dividing each nonzero entry of L by the sum of the entries in its row, and Lc the matrix which results by dividing each nonzero element of L by the sum of the entries in its column.

$$L_r = \begin{bmatrix} 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad L_c = \begin{bmatrix} 0 & 0 & 1/2 & 0 & 1/3 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 \end{bmatrix}$$

Then H consists of the nonzero rows and columns of $LrLc^T$, and A consists of the nonzero rows and columns of $Lc^TLr$.

$$L_rL_c^T = \begin{bmatrix} 5/12 & 0 & 2/12 & 0 & 3/12 & 2/12 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 3/4 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 1/3 \end{bmatrix} \quad L_c^TL_r = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/4 & 1/4 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/6 & 0 & 5/6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We ignore the rows and columns of A, H which consist entirely of zeros, since (by definition) all the nodes of G have at least one incident edge.

$$H = \begin{bmatrix} 5/12 & 0 & 2/12 & 3/12 & 2/12 \\ 0 & 1 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/4 & 0 & 0 & 3/4 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/2 & 1/4 & 1/4 \\ 0 & 1/2 & 1/2 & 0 \\ 0 & 1/6 & 0 & 5/6 \end{bmatrix}$$

Here are some properties of H and A. First, both matrices are primitive, since the Markov chains which they represent are aperiodic: when visiting any authority or hub, there is a probability to revisit it on the next entry to the authority or hub side of the bipartite graph. Second, the adjacency matrix of the graph of A is symmetric, since $a_{i,j} > 0$ implies $a_{j,i} > 0$. With the matrix we obtain here, we can detect two disconnected components from inspection, so we run the program on two different Markov Chains. The computation of SALSA on multiple irreducible components will be given in next section.

## 5.4 Markov Chains with Multiple Irreducible Components

In [Lempel et. al. 2000], we can see that sometimes both hub and authority of the SALSA algorithm has more than one strongly connected components. The chain formed by these components are irreducible respectively. Thus, each chain will have its only stationary distribution. Denote these components by $C_1$, $C_2$, . . . , $C_k$ where $C_i \in C$, 1 < i < k. Let e denote the |k| dimensional distribution vector, all whose entries equal 1/|k|.

For all vertices v in k, denote by c(v) the irreducible component to which v belongs, $c(v) = i \Leftrightarrow v \in A_i$. Let $\pi_1$, $\pi_2$, . . . , $\pi_k$ be the unique stationary distributions of the irreducible Markov chains induced by $C_1, \ldots, C_k$. Denote by $\pi_i^{c(v)}$ the entry which corresponds to v in $\pi_i^{c(v)}$. The random walk on $C_i$, $1 < i < k$, governed by the transition matrix P and started from all states with equal probability, will converge to a stationary distribution as follows:

$$\lim_{n\to\infty} eP_K^n = \pi \text{ where } \pi_i = \frac{|A_{c(v)}|}{|A|} \cdot \pi_i^{c(v)} \tag{5.4.1}$$

Since SALSA does not incorporate random reset, the graph structure affects whether the ranking vector is unique. If the bipartite graph H is not connected, then the authority and hub vectors are not unique. The original SALSA paper showed, using the ergodic theorem, that each component of the graph has a simple dominant eigenvalue which is equal to one. Each component will converge on its own, and then all result will depend on how much weight each component is given in the initialization.

We can see that there are two disconnected components in the bipartite graph, {B} and {A, C, E, F}. With uniform initial condition on each component, we will obtain the following result,

Table 5.1 The SALSA Score of Graph in Figure 5.1 with non-uniform initial distribution

| | | | | | | |
|---|---|---|---|---|---|---|
| Hub | 0.2667 | 0.2 | 0.1333 | 0 | 0.2667 | 0.1333 |
| Authority | 1 | 0 | 0 | 0 | 0 | 0 |

What we can find is only node A is considered as the only important authority node under the uniformed initial score. But apparently nodes C, D, E should all be authorities. This is clearly wrong.

Table 5.2 The SALSA Score of Graph in Figure 5.1 with uniform initial distribution

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| Hub | 0.2667 | 0.2 | 0.1333 | 0 | 0.2667 | 0.1333 |
| Authority | 0.25 | 0 | 0.3333 | 0.1667 | 0.5 | 0 |

This computation approach requires some overhead of graph traversal to find disconnected components [Cormen et. al. 2001]. Once the algorithm runs on billions of nodes, the time consumption will be enormous. Node A now is on longer considered as only important page with other nodes equally unimportant. The ranking vector tells us that it is important to decide what would be the appropriate initial distribution to execute SALSA.

Recall that in last chapter, we had HITS score on the same graph. SALSA algorithm to some extent is doing a better job in finding hubs and authorities. Notice that the HITS algorithm on Figure 5.1, has the authority rank (3 3 1 1 2 3) and hub rank (3 2 1 2 1 3). In fact, node A should be a decent authority since it has indegree of A, and F should be a good hub because it points to a good authority E. Now we can see that they are all rectified by SALSA algorithm. The authority rank and hub rank of these nodes has become (1 2 3 4 1 3) and (3 5 2 4 1 5). This is much more accurate for the give graph and important for finding the correct results for the according query.

## 5.5 SALSA on Irregular Graphs

In this section, we would like to give an analysis of SALSA on irrgular graph structures. We try to explore some details of the SALSA algorithm on small graphs.

For HITS and SALSA, there are some graphs that give rise to repeated eigenvalues. According to [Farahat et. al. 2006], when $L^TL$ or $LL^T$ of a graph has repeated dominant eigenvalues, the output of HITS algorithm is badly behaved. Let us have a test to further inspect if this theory hold for the case. In this example, we constructed a tree structure with node A has only incoming links and node D, E, F, G, H have only outgoing links.

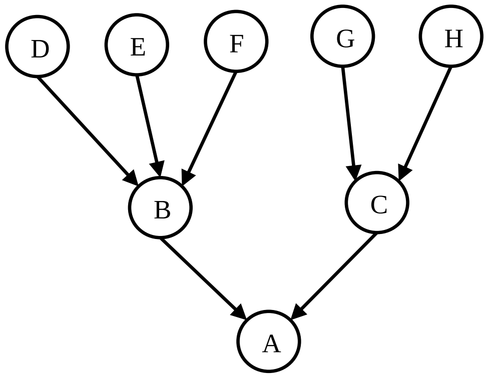


Figure 5.4 A Tree Structure Web Graph

Table 5.3 SALSA and HITS Score of Tree Structure

| SALSA | | HITS | |
|---|---|---|---|
| Hub Score | Authority Score | Hub Score | Authority Score |
| 0 | 0.3333 | 0 | 0 |
| 0.1428 | 0.3333 | 1 | 0 |
| 0.1428 | 0.3333 | 0 | 0 |
| 0.1428 | 0 | 0 | 0 |
| 0.1428 | 0 | 0 | 0 |
| 0.1428 | 0 | 0 | 0.5773 |

|  |  |  |  |
|---|---|---|---|
| 0.1428 | 0 | 0 | 0.5773 |
| 0.1428 | 0 | 0 | 0.5773 |

From the ranking score of two graphs, we find that while the eigenspaces of the adjacency matrix is {1, 1, 1, 0, 0, 0, 0}, and the HITS score is totally distorted. In SALSA ranking vector, we can find that nodes A, B, C are still considered good authorities and nodes B to H are still good hubs. We here conclude that despite the multiple set problem, SALSA is relatively stable toward graph changes. Actually, the reconstruction by SALSA has created several Markov Chains, thus they can avoid the problem caused by repeated dominant eigenvalue and less susceptible to irregular graph. In this way, SALSA is not quite sensitive in terms of the eigenvalues.

This reason of this badly behaved HITS score can be stated like this. $LL^T$ has a repeated largest eigenvalue, and the dominant eigenvector of $LL^T$ (and also $L^TL$) has zero entries for nodes whose out-degree (in-degree) is nonzero. If the hub or authority weight of node i converges to 0, then the corresponding entry from the dominant eigenvector must be 0. In fact, if the dominant eigenvalue is repeated, the score of these nodes will not be unique.

## 5.6 Performance Evaluation

We also conducted the same type of experiments as we did with HITS. The dataset we use are exactly the same as the one we used on HITS algorithm, and that way we can observe some relations between these two query-dependent algorithms.

Table 5.4 Top 10 authority results from computation geometry category with

SALSA

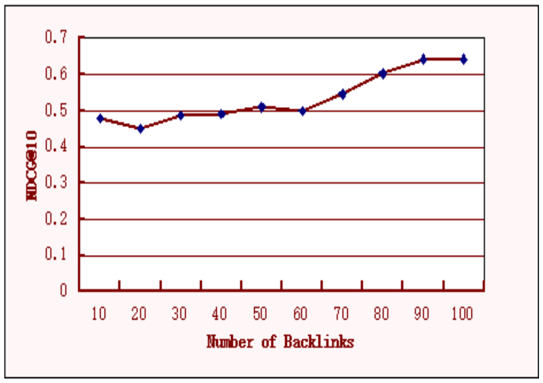
| |
|---|
| [R] http://www.eecs.tufts.edu/g/150GEO<br><br>COMP 150-GEO: Computational Geometry, Spring 2000 |
| [R] http://www.cs.uu.nl/CGAL<br><br>The former CGAL home page |
| [R] http://www.geom.umn.edu/software/cglist<br><br>Directory of Computational Geometry Software |
| [R] http://www.math.tau.ac.il/~sariel/CG.html<br><br>Computational Geometry Stuff |
| [O] http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html<br><br>Mesh Generation & Grid Generation on the Web |
| [R] http://compgeom.cs.uiuc.edu/~jeffe/compgeom/code.html<br><br>Computational Geometry Code |
| [R] http://www.laas.fr/~sleroy/Page2.html<br><br>Computational Geometry Issues |
| [R] http://link.springer.de/link/service/journals/00454<br><br>LINK: Peak-time overload |
| [R] http://www.cs.umn.edu/scg98<br><br>SCG 98 |
| [I]http://agnews.tamu.edu/~jpalmer/bookmarks/Computer_Graphics_Link_Collections_<br><br>index.html<br><br>Link Collections |

Table 5.5 Top 10 hub results from computation geometry category with SALSA

| |
|---|
| [I] http://dimacs.rutgers.edu/~istreinu/Reconnect99/software.html<br><br>Computational Geometry software bookmarks |
| [O] http://www.math.tau.ac.il/~sariel<br><br>Sariel's Home Page |
| [O] http://www.cs.cmu.edu/~quake/triangle.research.html<br><br>Triangle:    Research credit, references, and online papers |
| [O] http://www.acm.org/tog/resources/RTNews/html<br><br>Ray Tracing News Guide |
| [I] http://www.iwr.uni-heidelberg.de/iwr/comopt/mitarbeiter/simon/links.html<br><br>Links.html |
| [O] http://www.acm.org/pubs/citations/proceedings/compgeom/262839/p430-gartner<br><br>ACM Digital Library: Exact primitives for smallest enclosing ellipses |
| [O] http://jigsaw.w3.org/css-validator<br><br>W3C CSS Validation Service |
| [O] http://www.ecse.rpi.edu/Homepages/wrf<br><br>Wm. Randolph Franklin |
| [R] http://www.scs.carleton.ca/~csgs/resources/cg.html<br><br>Computational Geometry Resources |
| [I] http://sunsite.informatik.rwth-aachen.de/dblp/db/conf/compgeom<br><br>Computational Geometry |

If we compare the result of SALSA with that of HITS, it is clear that more pages in root set are returned. The property of bipartite graph will guarantee that if that page is a is pointed to by most of the pages in the graph, it will receive a higher ranking because of the random walk on such subgraph. There will be less similarity between authority pages and hub pages. Most of the authority pages come from the root set and most of the hub pages come from the base set that is pointing to the authority pages due to the computation of two independent chains

In our second experiment, separation between different domains is also applied to this test. SALSA is also query dependent algorithm. With the query University of Oklahoma, the same neighborhood graph can be used to execute the algorithm. We also use discounted cumulative gain as our measurement for SALSA.



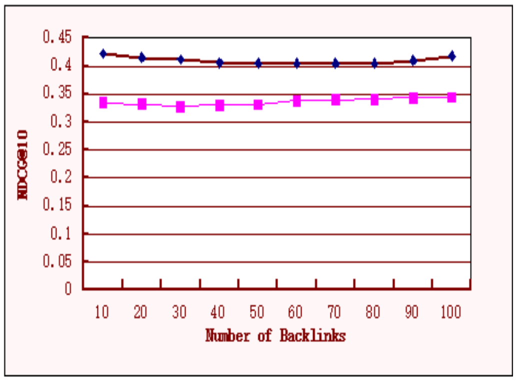
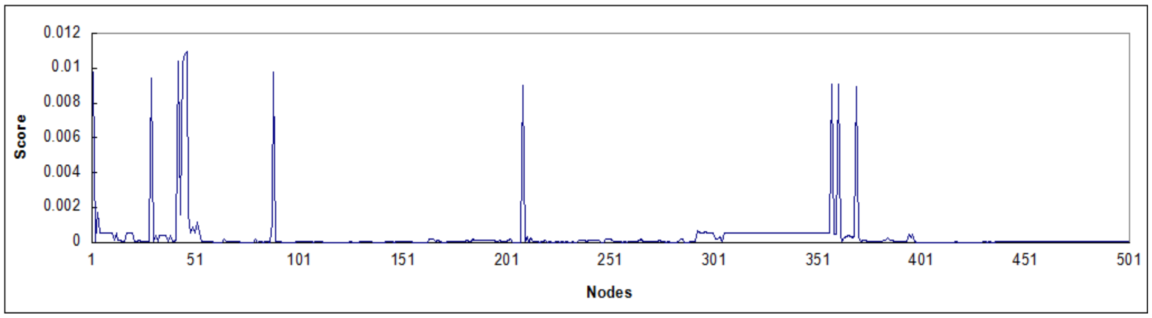


Figure 5.5 a) NDCG of SALSA in terms of query *University of Oklahoma* b) NDCG of SALSA in terms of query College of Engineering with and without domain separation

For SALSA algorithm, it could be to compare the result with HITS, as they obtained similar concept, except that SALSA uses random walk in its computation. As a result, we also find that it has the same property as the one in HITS that with the growing of the neighborhood graph, the search score will slightly get higher. That means SALSA has the same effect on the neighborhood graph and it will get stable if the main component of the graph does not change too much when we add more backlinks to the nodes.

One other thing we need to examine is the Tightly Knit Community (TKC) Effect according to [Lempel et. al. 2000]. A tightly knit community is a small but highly interconnected set of sites. This effect occurs when a community obtains a higher score, even though the pages in the TKC are not authoritative enough on the topic. The mutual reinforcement approach is vulnerable to this effect, and will sometimes rank the sites of a TKC in unjustified high positions.



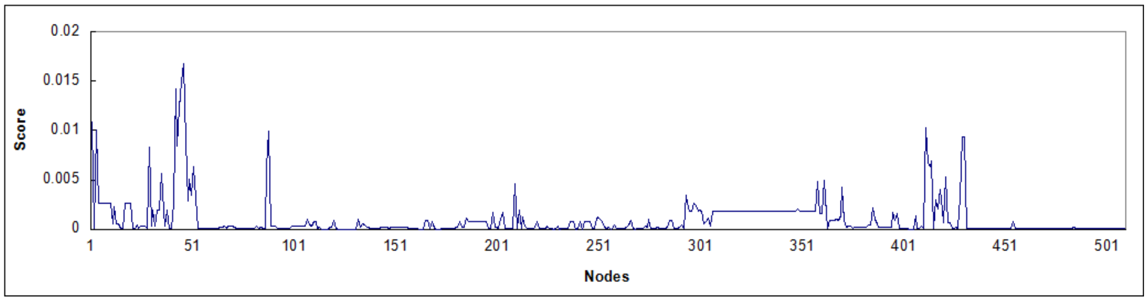
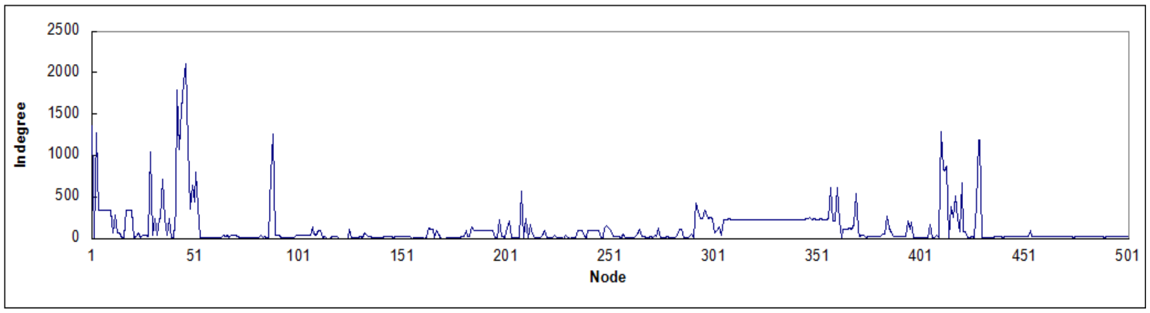


Figure 5.6 a) HITS score on query University of Oklahoma b) Degree Distribution of the first 500 nodes in this web graph



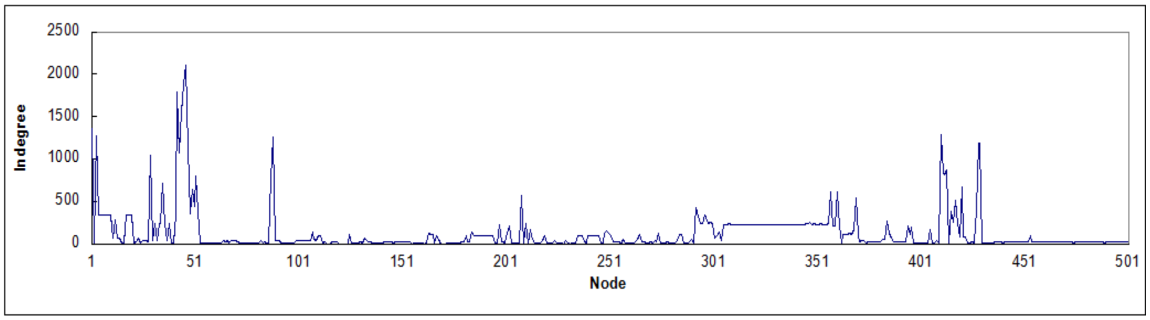


Figure 5.7 a) SALSA score on query University of Oklahoma b) Degree Distribution of the first 500 nodes in this web graph

Recall that in Figure 5.1, the authority ranking of the HITS algorithm is (3 3 1 1 2 3), and in SALSA ranking, it became (3 5 2 4 1 5). This gives an intuition that to some extent, SALSA has the ability to overcome TKC effect. On a graph with relatively large size, HITS failed to identify some authority nodes. But on the same web graph, as we see in Figure 5.6 and Figure 5.7, the effect is also straightforward. Most of the nodes with high indegree received a corresponding score and identified as good authorities. Compared to HITS, SALSA is more likely to follow the degree distribution, in comparison, HITS gives nodes a high score when they obtain high indegree and that tend to ignore the ranking for some decent pages. Especially, if these pages does not have a dominant impact on the graph.

## 5.7 Conclusion

SALSA incorporates the properties of both PageRank and HITS. So it is clear that SALSA has hubs and authorities as mutual reinforcements as well as derives Markov Chain theorem.

One big leap of SALSA is less susceptible to topic drift and link spamming, because the coupling of hub and authority matrix is less strict. However, the drawback of SALSA is more or less the same as that of HITS since it requires a pre-constructed neighborhood graph. Then at query time, two Markov Chains have to be formed. Moreover, SALSA algorithm may not be unique since it does not force irreducibility.

SALSA algorithm may not be a proper approach for universal query system, but just like HITS, it is suitable for query from local community as the recommendation system of Twitter[Gupta et. al. 2013]. Mutual Reinforcement Relationship has a better performance within a dense community. SALSA has a prominent effect on dealing with topic drift proble, which significantly improved the performance of ranking.

# Chapter 6

# Conclusion

In the thesis, we discussed about three major algorithms for modern information retrieval. We analyzed their strength and weakness respectively in terms of the mathematical properties and performance.

Currently, there are a large amount of analysis of those algorithms based on their mathematical properties, whereas few resource could be found about their performance on large scale data sets. So it is necessary for us to focus more on the evaluation of their performance in different situations.

From the analysis in the thesis, it is clear that although these algorithms are widely used in different areas, there could still be some improvements on them with respect to different applications. With their unique properties, these algorithms should be considered differently when we want to implement such technique. As they have become mature in most of the area, whether they are efficient in all the areas they have been applied to are not assured.

One of the future work is to explore their performance more on different applications other than web search regarding various type of structures and data sets. Other would be exploit some unique properties of these algorithms and find solutions in order to improve the performance.

# References

Adamic L. A. and B. A. Huberman. Zipf's law and the Internet. Glottometrics, 3. 2002.

Andersen R., F. Chung, and K. Lang. Local graph partitioning using PageRank vectors. In Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science. 2006.

Baeza-Yates, R. and B. Ribeiro-Neto. Modern Information Retrieval. ACM Press, New York, 1999

Borodin, A., G. O. Roberts, J. S. Rosenthal. Link analysis ranking: algorithms, theory, and experiments. ACM Transactions on Internet Technology. 2005

Brin, S. and L. Page, The anatomy of a large-scale hypertextual Web search engine, Proc. 7th Int. WWW Conference. 1998.

Broder, A., R. Kumar, F. Maqhoul, P. Raghavan, S. Pajagopalan, R. Stata, A. Tomkins, and J.Wiener. Graph Structure in the Web. Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking. Pages 309-320. 2000

Chung, F.. A Brief Survey of PageRank Algorithms.IEEE Transactions On Network Science And Engineering, Vol. 1, No. 1. 2014

Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein. Introduction to Algorithms. MIT Press. 2001.

Duan, Y., J. Wang, M. Kam, and J. Canny, A Secure Online Algorithm for Link Analysis on Weighted Graph. Proceedings of the Workshop on Link Analysis, Counter-terrorism, and Security (in conj. with SIAM International Conference on Data Mining), Newport Beach, CA, USA, Pages 71–81. 2005

Easley, D. and J. Kleinberg. Networks, Crowds, and Markets: Reasoning about a Highly Connected World. Cambridge University Press, 2010

Estrada, E.. The Structure of Complex Networks: Theory and Applications, Oxford University Press, 2011

Farahat, A., T. Lofaro, J. C. Miller, and G. Rae, L. A. Ward. Authority Rankings from HITS, PageRank, and SALSA: Existence, Uniqueness, and Effect of Initialization. SIAM Journal On Scientific Computing, Vol. 27, No. 4, 2006

Fetterly, D., N. Craswell, and V. Vinay. Search effectiveness with a breadth-first crawl. Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, Pages 755-756. 2008

Gleich, D. F. PageRank beyond the Web. SIAM Review, Volume. 57, No. 3, Pages 321–363. 2014

Gupta, P., A. Goel, J. Lin, A. Sharma, D. Wang, and Reza Zadeh. WTF: The Who to Follow Service at Twitter. Twitter, Inc. 2013

Haveliwala, T.. Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Transactions on Knowledge and Data Engineering. 2003

Ipsen, Ilse C.F. and Rebecca S. Wills. Mathematical properties and analysis of google's pagerank. Bol. Soc. Esp. Mat. Apl., 34:191–196. 2006.

Järvelin, K., J. Kekäläinen . Cumulated Gain-Based Indicators Of IR Performance. ACM Transactions on Information Systems (TOIS) Volume 20 Issue 4, Pages 422-446. 2002.

Kleinberg, J., Authoritative sources in a hyper-linked environment, Proc. 9th ACM–SIAM Symposium on Discrete Algorithms. 1998.

Kleinberg, J. and S. Lawrence. The Structure of the Science Web.
Science: 294 (5548), 1849-1850. 2001

Lee, H. C., A. Borodin. Perturbation of the Hyper-Linked Environment. 9th Annual International Conference, COCOON 2003 Big Sky, MT, USA. 2003.

Lempel, R. and S. Moran. The stochastic approach for link-structure analysis and the TKC effect. In the Ninth International World Wide Web Conference, New York, ACM Press. 2000.

Lempel, R. and S. Moran, SALSA: The Stochastic Approach for Link-Structure Analysis. ACM Transactions on Information Systems, Vol. 19, No. 2, April 2001, Pages 131–160. 2000.

Langville, A. and Carl D. Meyer. Deep inside PageRank. Internet Mathematics Journal, 1(3):335-80. 2005.
Langville, A. and C. D. Meyer. Google's PageRank and beyond: the science of search engine rankings. Princeton University Press. 2006.

Najork, M.. Comparing the effectiveness of hits and salsa. Proceedings of the sixteenth ACM conference on Conference on information and knowledge management, Pages 157-164. 2007.

Najork, M., S. Gollapudi, and R. Panigrahy. Less is more: sampling the neighborhood graph makes SALSA better and faster Web Search and Data Mining - WSDM , pp. 242-251. 2009.

Najork, M. A., H. Zaragoza, and M. J. Taylor, HITS on the Web: How does it Compare?. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, Pages 471-478, 2007.

Newman, M. E. J., The structure and function of complex networks. SIAM REVIEW, 45(2), Pages 167–256. 2003

Ng, A. Y., A. X. Zheng, and M. I. Jordan. Link analysis, eigenvectors and stability. Proceedings of the 17th international joint conference on Artificial intelligence, Volume 2, Pages 903-910. 2001.

Page, L., S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking:Bringing order to the web. Tech. Rep. Computer Systems Laboratory, Stanford University, Stanford, CA, 1999.

Salton G., A. Wong and C.S. Yang. A vector space model for automatic indexing. Communications of the ACM 18 (11), Pages 613–620. 1975.

Singhal, A. Modern Information Retrieval: A Brief Overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43. 2001

Stephen, R. and Z. Hugo. The Probabilistic Relevance Framework: BM25 and Beyond. Foundations and Trends□ in Information Retrieval Vol. 3, No. 4, Pages 333–389. 2009.

Wang, G. A., J. Jiao, A. S. Abrahams, W. Fan, and Z. Zhang, ExpertRank: A topic-aware expert finding algorithm for online knowledge communities. Decision Support Systems, Volume 54, Issue 3. 2013

Zaragoza, H., N. Craswell, M. Taylor, S. Saria, and S. Robertson. Microsoft Cambridge at TREC-13: Web and HARD tracks. In Proceedings of TREC. 2004.

# Appendix A

This Appendix gives a detail proof of some theorem we mention in the thesis including the convergence of PageRank and HITS, as well as the eigenspace of the stochastic matrix of PageRank.

We would like to give a proof of the rate of convergence of the power method in the process of PageRank computation.

$$\pi_k = \alpha_1\lambda_1^k v_1 + \alpha_2\lambda_2^k v_2 + \cdots + \alpha_n\lambda_n^k v_n$$

$$= A\cdot(\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n)$$

$$\pi_1 = \alpha_1 A v_1 + \alpha_2 A v_2 + \cdots + \alpha_n A v_n$$

$$= \alpha_1\lambda_1 v_1 + \alpha_2\lambda_2 v_2 + \cdots + \alpha_n\lambda_n v_n$$

$$\pi_k = \alpha_1\lambda_1^k v_1 + \alpha_2\lambda_2^k v_2 + \cdots + \alpha_n\lambda_n^k v_n$$

$$= \alpha_1\lambda_1^k (v_1 + \alpha_2\left(\lambda_2/\lambda_1\right)^k v_2 + \cdots + \alpha_n\left(\lambda_n/\lambda_1\right)^k v_n)$$

Next, we want to show the convergence of HITS algorithm. We give a proof for hub side, and the authority side will be exactly the same.

We have mentioned that $LL^T$ is symmetric. Let's write the resulting mutually orthogonal eigenvectors of $LL^T$ as $z_1$, $z_2$, . . . , $z_n$, with corresponding eigenvalues $\lambda_1$, $\lambda_2$, . . . , $\lambda_n$ respectively, and let's order the eigenvalues so that $|\lambda_1| > |\lambda_2| > \cdot\cdot\cdot > |\lambda_n|$. Furthermore, to make things simpler in this explanation, let's suppose that $|\lambda_1| > |\lambda_2|$. Now, given any vector x, a good way to think about the matrix-vector product $(LL^T)x$ is to first write x as a linear combination of the vectors z1, . . . , zn. With coefficients $a_1, \dots, a_n$, $x = a_1z_1 + a_2z_2 + \cdots + a_nz_n$, and we can derive the formula $a_1\lambda_1z_1 + a_2\lambda_2z_2 + \cdots + a_n\lambda_nz_n$ from $(LL^T)x$, from where the equality follows the fact that each vi is an eigenvector. When the vector x multiplies repeatedly by $LL^T$, we have the result of $(LL^T)x$ to the power of k, and that gives us $(LL^T)^k x = a_1\lambda_1^k z_1 + a_2\lambda_2^k z_2 + \cdots + a_n\lambda_n^k z_n$.

Let's think of this in the context of the vectors of hub scores, where $x_k = (LL^T)x_0$. Recall that x0 is just the starting vector of all 1's, it can be represented in terms of the basis vectors z1, . . . , zn as some linear combination. So with $x_0 = a_1z_1 + a_2z_2 + \cdots + a_nz_n$

$$x^k = a_1(LL^T\lambda_1)^k z_1 + a_2(LL^T\lambda_2)^k z_2 + \cdots + a_n(LL^T\lambda_n)^k z_n$$

and if we divide both sides by $\lambda_1^k$ , then we get

$$x^k \big/ \lambda_1^k = (LL^T)^k[a_1z_1 + a_2\left(\lambda_2 \big/ \lambda_1\right)^k z_2 + \cdots + a_n\left(\lambda_n \big/ \lambda_1\right)^k z_n]$$

Recall our assumption that $|\lambda_1| > |\lambda_2|$, we see that as k goes to infinity, every term on the right-hand side but the first is going to 0. As a result, the sequence of vectors is converging to the limit $a_1z_1$ as k goes to infinity. So the HITS with normalization always converges, and the rate of convergence is given by

$$[\lambda_2(L^TL)/\lambda_1(L^TL)]^k.$$

Then, we would like to see the property of Google Matrix G. In another word, the relation of the spectrum of G and stochastic matrix S.

S is stochastic, (1, e) is an eigenpair of S. Let Q = (e X) be a non-singular matrix that has the eigenvector e as its first column.

Let $Q^{-1} = \begin{pmatrix} y^T \\ Y^T \end{pmatrix}$, then $Q^{-1}Q = \begin{pmatrix} y^T e & y^T X \\ Y^T e & Y^T X \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & I \end{pmatrix}$.

It gives some identities $y^T e = 1$ and $Y^T e = 0$. Thus, the transformation will be $Q^{-1}SQ = \begin{pmatrix} y^{-1}e & y^T SX \\ Y^T e & Y^T SX \end{pmatrix} = \begin{pmatrix} 1 & y^T SX \\ 0 & Y^T SX \end{pmatrix}$, which reveals that $Y^T SX$ contains the remaining eigenvalues of S. Apply the transformation to $G = \alpha S + (1-\alpha)ev^T$ gives

$$\begin{aligned}
&Q^{-1}(\alpha S + (1-\alpha)ev^T)Q \\
&= \alpha Q^{-1}SQ + (1-\alpha)Q^{-1}ev^T Q \\
&= \begin{pmatrix} \alpha & \alpha y^T SX \\ 0 & \alpha Y^T SX \end{pmatrix} + (1-\alpha)\begin{pmatrix} y^T e \\ Y^T e \end{pmatrix}\begin{pmatrix} v^T e & v^T X \end{pmatrix} \\
&= \begin{pmatrix} \alpha & \alpha y^T SX \\ 0 & \alpha Y^T SX \end{pmatrix} + \begin{pmatrix} 1-\alpha & (1-\alpha)v^T X \\ 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 & \alpha y^T SX + (1-\alpha)v^T X \\ 0 & \alpha Y^T SX \end{pmatrix}
\end{aligned}$$

Therefore, the eigenvalues of $G = \alpha S + (1-\alpha)ev^T$ are $\{1,\ \alpha\lambda_1,\ \ldots,\ \alpha\lambda_n\}$.

If $\pi^T(\alpha) = (\pi_1(\alpha), \pi_2(\alpha), ..., \pi_n(\alpha))$ is the PageRank vector, then

$\left| \frac{d\pi_j(\alpha)}{d\alpha} \right| \le \frac{1}{1-\alpha}$, for each j = 1, 2, ..., n,

*Proof.* Compute the difference of both sides of $\pi^T(\alpha) = \pi^T(\alpha)(\alpha S + (1-\alpha)ev^T)$ and that yields $\frac{d\pi^T(\alpha)}{d\alpha}(I - \alpha S) = \pi^T(\alpha)(S - ev^T)$. Matrix I-αS(α) is nonsingular because α<1, so $\frac{d\pi^T(\alpha)}{d\alpha} = \pi^T(\alpha)(S - ev^T)(I - \alpha S)^{-1}$.

For every real $x \in e^{\perp}$ and for all real vector $y_{n\times 1}$,

$$\left| x^T y \right| = \|x\|_1 \left(\frac{y_{\max} - y_{\min}}{2}\right).$$

This is the consequence of Holder's inequality because for all real α,

$$\left|x^T y\right| = \left\|x^T (y - \alpha e)\right\| \le \|x\|_1 \|y - \alpha e\|_\infty ,$$

so $\min_\alpha \|y - \alpha e\|_\infty = (y_{\max} - y_{\min}) / 2,$ where minimum is attained when $\alpha = (y_{\max} + y_{\min}) / 2.$ Then we can derive

$$\frac{d\pi_j(\alpha)}{d\alpha} = \pi^T(\alpha)(S - ev^T)(I - \alpha S)^{-1} e_j ,$$

Ej is the jth standard basis vector. Since $\pi^T(\alpha)(S - ev^T)e = 0,$ the inequality is applied with $y = (I - \alpha S)^{-1} e_j$ to obtain

$$\left|\frac{d\pi_j(\alpha)}{d(\alpha)}\right| \le \left\|\pi^T(\alpha)(S - ev^T)\right\|_1 \left(\frac{y_{\max} - y_{\min}}{2}\right).$$

As $\left\|\pi^T(\alpha)(S - ev^T)\right\|_1 \le 2,$ so $(I - \alpha S)^{-1} \ge 0,$

$$\left|\frac{d\pi_j(\alpha)}{d\alpha}\right| \le y_{\max} - y_{\min} .$$

Now we use the fact that $(I - \alpha S)^{-1} \ge 0,$ we can conclude that

$$y_{\max} \le \max_{i,j} [(I - \alpha S)^{-1}]_{ij} \le \left\|(I - \alpha S)^{-1}\right\|_\infty = \left\|(I - \alpha S)^{-1} e\right\|_\infty = \frac{1}{1 - \alpha} .$$

Then consequently,

$$\|\pi - \pi'\|_1 \le \frac{2}{1 - \alpha} .$$

Suppose $G = \alpha S + (1 - \alpha) ev^T$ is the Google matrix with PageRank vector $\pi^T$ and $G' = \alpha S' + (1 - \alpha) ev^T$ is updated Google matrix with corresponding PageRank vector $\pi'$. Then

$$\left\|\pi^T - \pi'^T\right\|_1 \le \frac{2\alpha}{1-\alpha}.$$

*Proof.* Let F be the matrix representing the perturbation between the two stochastic matrices S and S’. Thus, F=S-S’. Then

$$\begin{aligned}\pi^T - \pi'^T &= \alpha\pi^T S - \alpha\pi'^T S' \\ &= \alpha\pi^T S - \alpha(\pi'^T - \pi^T + \pi^T)S' \\ &= \alpha\pi^T S - \alpha\pi^T S + \alpha(\pi^T - \pi'^T)S' \\ &= \alpha\pi^T F + \alpha(\pi^T - \pi'^T)S' \\ &= \alpha\pi^T F(I - \alpha S')^{-1}.\end{aligned}$$

Computing norms will give us

$$\left\|\pi^T - \pi'^T\right\|_1 \le \alpha\left\|\pi^T F\right\|_1\left\|(I-\alpha S')^{-1}\right\|_\infty = \frac{\alpha}{1-\alpha}\left\|\pi^T F\right\|_1$$

Reorder F so that the rows corresponding to updated pages are at top of the matrix. Then,

$$\pi^T F = (\pi_1^T \quad \pi_2^T)\begin{pmatrix} F_1 \\ 0 \end{pmatrix} = \pi_1^T F_1.$$

Therefore, $\left\|\pi^T F\right\|_1 = \left\|\pi_1^T F_1\right\|_1 \le \left\|\pi_1^T\right\|_1 \left\|F_1\right\|_\infty$. And $\left\|F_1\right\|_\infty = \left\|S_1 - S_1^{'}\right\|_\infty \le \left\|S_1\right\|_\infty + \left\|S_1^{'}\right\|_\infty = 2$, where $S_1$ and $S_2$ also correspond to the updated pages. Therefore $\left\|\pi^T F\right\|_1 \le 2$. And finally,

$$\left\|\pi^T - \pi'^T\right\|_1 \le \frac{2\alpha}{1-\alpha}.$$