OPTIMISATION OF PAGERANK ALGORITHM IN HADOOP

Dissertation

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF TECHNOLOGY
IN
SOFTWARE TECHNOLOGY
BY

PULKIT GOEL
(Roll No: 2K13/SWT/012)
Reg.No: DTU/13/M.TECH/489

MAJOR PROJECT REPORT II
(Paper Code: CO 821)

Under the guidance of
Dr. Rajni Jindal (DTU)

SHAHBAD DAULATPUR, MAIN BAWANA ROAD, NEW DELHI, DELHI 110042
INDIA

DELHI TECHNOLOGICAL UNIVERSITY
NEW DELHI

STUDENT DECLARATION

I hereby declare and undertake that this submission is my original experiments with PageRank algorithm and to the best of my information and I believe, it contains no text material previously published or written by another person nor text material which has been accepted for the award of any other degree or diploma of any Institute or any other University of higher education and learning, except where due acknowledgement has been made in the text report. Project works associated to this report are well discussed and improved under the **Dr. Rajni Jindal** (Mentor) supervision from **DTU (Delhi Technical University)**.

DATE:

SIGNATURE:

PULKIT GOEL
ROLL NO: 2K13/SWT/012
REGISTRATION NO: DTU/13/M.TECH/489

DELHI TECHNOLOGICAL UNIVERSITY
NEW DELHI

CERTIFICATE

This is to certify that the project entitled, "**OPTIMISATION OF PAGERANK ALGORITHM IN HADOOP**", has been carried out by him under my supervision and guidance in partial fulfillment of the degree of Master of technology in Computer Science & Engineering at DTU, New Delhi during the academic year 2016.

Date:

Dr. Rajni Jindal
Department of Computer Science and Engineering
Delhi Technological University

ACKNOWLEDGEMENT

I am presenting my project report on title "**OPTIMISATION OF PAGERANK ALGORITHM IN HADOOP"** with a lot of pleasure and satisfaction. I take this opportunity to thank my supervisor, **Dr. Rajni Jindal**, for guiding me and providing me with all the guidance and facilities, which paved way to the successful completion of this project. I am also very thankful to my friends and colleagues who have been a source of encouragement and inspiration throughout the duration of this project work. I would like to express my deep gratitude to my parents. Their continuous love and support gave me strength for pursuing my dream. I am also thankful to the **SAMSUNG RESEARCH INSTITUTE, Noida** who has provided me opportunity to take admission in the **M.Tech**. Programme and to gain my knowledge through this programme. This curriculum provided me knowledge and opportunity to grow in various domains of computer science and Software Technology. At last, I am thankful to the faculty members (Dr. Rajni Jindal, Dr. O.P. Verma(HOD)) from **DTU (Delhi Technical University),** who visited the Samsung premises to guide and teach. Their knowledge and efforts helped me to grow and learn in the field of computer science and software technology. I feel very proud that their contribution helped me to bring out new ideas in my professional life.

PULKIT GOEL
ROLL NO: 2K13/SWT/012
REGISTRATION NO: DTU/13/M.TECH/489

Contents

ABSTRACT

PageRank was an algorithm used by Google in Searches to rank all websites in its search engine results. PageRank algorithm works by counting number and quality of web links to a page to determine an estimate of how important the website. The underlying assumption is more important websites are likely to receive more links from other websites. It is not the only algorithm used by Google Company to order search engine results, but it is the first type of algorithm that is used by the company. Modification to this PageRank has been presented So that it can be fast running time using Hadoop and good search results for a user as we get the good page rank value.

Literature Review/Motivation

In the past few years, Google Inc. has become the most used web search engine in the world. The main purpose of this project is to provide all aspects of PageRank. The contents of PageRank Algorithm primarily right upon papers by Google founders Lawrence Page and Sergey Brin at the time of graduation at Stanford University USA.

Introduction of Google PageRank

Completive factor behind Google Inc. search engine was high Run time performance and the superior quality and ease of use, when search results compared with other search engines like Bing, Yahoo. These quality and value of search results is based on PageRank value, a method to rank the web documents.

It is argued that, especially considering the dynamic of the internet, too much time has passed since the mathematics and scientific work is carried out on PageRank, as that it is still could be the basis of the ranking methods in the Google search engine. There is no doubt that from the past years many changes, adjustments and modifications regarding the ranking methods of Google have taken place, but PageRank was absolutely main role for Google's success, so at least fundamental concept behind PageRank should still be constant.

PageRank Concept

In early stages of the WWW (World Wide Web), search engines developed totally different strategies to rank the net pages. Nowadays, the incidence of a groundwork phrase inside document is one in every of the main factors inside the ranking techniques of any computer program. The occurrences of any search phrase will thereby be weighted by the length of a document (ranking by keyword density) or by its accentuation inside a document by markup language tags.

For the aim of higher search results and to form search engines resistant against mechanically (dynamic) generated internet, the conception of link quality was developed. This idea says, the amount of incoming links for a document measures its importance. Hence, an online page is a lot of vital, if several different web content link thereto (Forward Link). The conception of link quality avoids smart rankings most of times for the pages that square measure created to deceive search engines and that haven't got any significance inside the net, however varied webmasters elude it by making lots of incoming links for entranceway pages from even as insignificant different web content.

Contrary to the conception of link quality, PageRank isn't solely primarily based upon the entire range of incoming links. The essential approach of PageRank is

that: A document is taken into account the lot of vital if the lot of different documents link thereto, however those incoming links don't count equally. A document that ranks higher in terms of PageRank worth, if different high page ranking worth documents link there to.
So, inside this PageRank conception, the rank of a document is given by the total of rank of these documents that link thereto. And their rank once more is given by the worth of rank of documents that link to them. Hence, the PageRank of a document is usually determined reiterative recursively by the PageRank of different documents. Rank of any document influences or amendment the rank of the other page, PageRank is, in the end, primarily based upon the linking structure of the entire World Wide internet. Though this approach appears to be terribly broad and sophisticated in term of structure, Page and Brin (Google Cofounder) were ready to place this into Equations written by simple Mathematical formula.

PageRank Algorithm

The original PageRank algorithmic rule is delineated by Lawrence Page and Sergey Brin (Google Cofounder) in many publications. It's given by:

$$PR('A') = (1-d) + d\,(PR('T1')/C('T1') + \dots + PR('Tn')/C('Tn'))$$

where

- PR('A') is the PageRank of page 'A',
- PR('Ti') is the PageRank of pages 'Ti' which link to page 'A',
- C('Ti') is the number of outbound links on page 'Ti' and
- 'd' is a damping factor which can be set between '0' and '1'.

The PageRank of pages 'Ti' that links to page 'A' doesn't influence the PageRank of page 'A' uniformly. inside the PageRank algorithmic rule, the PageRank of a page 'Ti' is usually weighted (or divided) by the quantity of outward-bound links C('Ti') on page T. which means that the a lot of outward-bound links a page 'Ti' has, the less are going to be enjoy to Page 'A' from that link thereto on page T.

So, we are able to see that PageRank doesn't rank internet sites as a full, however it's determined for every page separately. Moreover, this PageRank of page A is recursively outlined by the PageRank's of these pages that directly link to page 'A'.

This weighted PageRank of pages 'Ti' is then additional up. The result of this is often that an additional incoming link to page 'A' can continuously increase page A's PageRank.

Finally, We add of the weighted PageRank's for all pages 'Ti' is increased with a damping issue (d) whose worth is set between '0' and '1'. Thereby, the extension of good thing about PageRank from a page by another page linking thereto is reduced.

Random Web Surfer(Swimmer) Model

In this publication, Lawrence Page and Sergey Brin (Google founder) provides a terribly easy intuitive justification on PageRank algorithmic program. They take into account PageRank as a model of user behavior, once a web swimmer clicks on links indiscriminately with no regard towards content.

When a random swimmer visits an internet page with a particular chance, the chance that the random swimmer can clicks on one link is entirely given by the amount of links there are on the page. This is often why one page's PageRank

isn't utterly passed on to a page it links thereto, however it's divided by the amount of links on the page.
So, the chance for the random swimmer reaching one page by clicking indiscriminately is that the total of chances for the random swimmer following links to the present page. Now, this chance is reduced by the damping issue d. this is often the justification among the Random web swimmer Model, therefore, is that the swimmer doesn't click on associate degree infinite range of links, however gets bored in typically and jumps to a different page indiscriminately.
The chance for the random swimmer for not stopping to click on the links is given by the damping issue d, which is, betting on degree of chance thus, set between '0' and '1'. The upper the (damping factor)**'d'** is, the a lot of doubtless can random swimmer keep clicking on links. Since the online swimmer jumps to a different page every which way solely when he stopped clicking links, therefore, the chance is enforced as a relentless (1-d) within the algorithmic program. Regardless range of inward links, the chance for the random net swimmer jumping to a page is usually (1-d), thus a page continuously contains a minimum PageRank price.

A Different Notation of PageRank Algorithm

Lawrence Page and Sergey Brin (Google Cofounders) revealed 2 totally different versions of their PageRank rule in several papers. Within the second version of the rule, the PageRank of page A is given as:

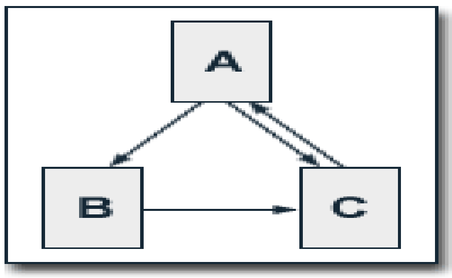
$$PR('A') = (1-d) / N + d\,(PR('T1')/C('T1') + \ldots + PR('Tn')/C('Tn'))$$

Where **'N'** is that the total **Numbers** of all pages on the **net**. The second version of this rule, indeed, doesn't a lot of disagree essentially from the primary one. **Concerning** the Random Internet Swimmer Model, the second version of PageRank of a page is that the actual chance for a random Swimmer reaching that page once clicking on several links. therefore PageRank's value is a kind of chance distribution over web content, therefore the total add of all pages' of the online, PageRank are one.
Contrary, within the initial version of this rule the chance for the random internet Swimmer reaching a page is weighted by the whole variety of (N) web content. So, during this version of PageRank is associate degree mean for the random Swimmer visiting that page, once he restarts this procedure as usually because the internet of has those (N) pages. If an online had a hundred pages

and a page incorporates a PageRank worth two, then random surfboarder may reach that page in a median double, if he restarts a hundred times.
As mentioned higher in example, the 2 versions of this rule don't disagree essentially from one another. A PageRank, that has been calculated by victimization the second version of that rule, should be increased by the whole variety of (N) web content to induce the according PageRank that might are calculated by victimization of the primary version. Even Page and Brin required the 2 versions of algorithms in their most well liked paper "The Anatomy of a Large-Scale Hyper matter net Search Engine", wherever they claim the primary version of this rule to create a chance distribution over web content with the intuitions of add of all pages' PageRank's being one.
In the following Project Report, we'll use the primary version of the rule. the explanation is that, PageRank calculations by suggests that of this rule as easier to work out, as a result of we will ignore the whole variety of web content, as we tend to don't grasp the worth of N.

The Characteristics of PageRank Algorithm

The characteristic of PageRank is illustrated by a little example.



We take into account a little internet, consisting of 3 pages 'A', 'B' and 'C' solely, whereby page 'A' links to the pages 'B' and 'C', page 'B' links to page 'C' and page 'C' links to page 'A'. In line with Page and Brin (Cofounders of Google), the damping issue (d) is often set to '0.85', however to stay the calculation straightforward mathematically we tend to set this to '0.5'. the precise price of the damping issue (d) has effects on PageRank worth, however it doesn't influence the principles of PageRank. So, we are going to get the subsequent equations for this PageRank calculation:

$$PR('A') = 0.5 + 0.5\,PR('C')$$
$$PR('B') = 0.5 + 0.5\,(PR('A') / 2)$$
$$PR('C') = 0.5 + 0.5\,(PR('A') / 2 + PR('B'))$$

These equations will simply be mathematically solved manually. We tend to get the subsequent PageRank price for the pages:

$$PR('A') = 14/13 = 1.07692308$$
$$PR('B') = 10/13 = 0.76923077$$
$$PR('C') = 15/13 = 1.15384615$$

It is obvious that this can be add of all pages PageRank’s is ‘3’ and so equals the overall variety of (N) sites is additionally ‘3’. As shown on top of, this can be a selected result for our easy example.
For this three-page example it's straightforward to resolve the equation system to see PageRank values. In follow, the **web** consists of billions countless documents and this can be out of the question to seek out an answer by this examination.

The Iterative Computation of the PageRank

Because of the scale of the initial internet, the Google program uses approximate, reiterative arithmetic computation of the PageRank values. this implies that, every page is assigned a random initial beginning worth PageRank value and also the PageRank’s of all websites is then calculated in many mathematical computation cycles supported by these equations determined by the PageRank algorithmic program. This reiterative calculation shall once more be illustrated for our same three-page example, whereby presumptuous every page is assigned a beginning PageRank worth of ‘1’.

Iteration	PR('A')	PR('B')	PR('C')
'0'	1	1	1
'1'	1	0.75	1.125
'2'	1.0625	0.765625	1.1484375
'3'	1.07421875	0.76855469	1.15283203
'4'	1.07641602	0.76910400	1.15365601
'5'	1.07682800	0.76920700	1.15381050
'6'	1.07690525	0.76922631	1.15383947
'7'	1.07691973	0.76922993	1.15384490
'8'	1.07692245	0.76923061	1.15384592
'9'	1.07692296	0.76923074	1.15384611
'10'	1.07692305	0.76923076	1.15384615
'11'	1.07692307	0.76923077	1.15384615
'12'	1.07692308	0.76923077	1.15384615

We can see, we have a tendency to get decent approximation results of the important PageRank values once solely '12' iterations in keeping with publications by Lawrence Page and Sergey Brin, concerning total '100' iterations are necessary to urge a decent approximation of the PageRank values for the total internet.
Also, by means that of this reiterative calculation, the total of all pages' PageRank's converges to the entire variety of (N) websites. therefore the average PageRank of an online page is usually '1'. There's minimum PageRank of a page that is usually given by (1-d). Therefore, there's a most PageRank of a page that is given by d(N)+(1-d), wherever (N) is total variety of websites on an internet. This most worth will in theory occur, if providing all web content exclusively link to at least one page solely, and this page additionally exclusively links to itself additionally.

Effect of Inbound Links

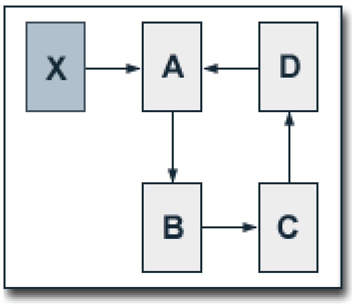
It has been already shown that every further inward link of an internet page continuously will increase the page's PageRank. Taking a glance at this PageRank formula, this can be given by:

$$PR('A') = (1-d) + d\,(PR('T1')/C('T1') + \ldots + PR('Tn')/C('Tn'))$$

One could assume that an additional incoming link from page X will increase the PageRank of page A by:

= d * PR('X') / C('X')

Where PR('X') is that the PageRank of page 'X' on the online and C('X') is that the total variety of its outward-bound links from page 'X'. However page 'A' typically links to alternative pages itself. Thus, these pages additionally get a PageRank have the benefit of this. If these pages back link to page 'A', then page 'A' can have a fair higher PageRank from its extra arriving link.
These single effects of extra arriving links shall be illustrated by this instance.



In this regard an internet site consisting of 4 pages 'A', 'B',' C' and 'D' that is connected to every different and kind in a circle manner. While not an external incoming links, we tend to currently add a page 'X' to the current example, for this we tend to assume a continuing PageRank PR('X') of '10'. Further, if page 'X', links to page 'A', by its only 1 outward-bound link. Considering the damping issue (d) to '0.5', we are going to get the subsequent equations for the PageRank values of the only pass of our site:

PR('A') = 0.5 + 0.5 (PR('X') + PR('D')) = 5.5 + 0.5 PR('D')
PR('B') = 0.5 + 0.5 PR('A')
PR('C') = 0.5 + 0.5 PR('B')
PR('D') = 0.5 + 0.5 PR('C')

Since, the full variety of outward-bound links for each page is one; this outward-bound links doesn't thought to be thought-about during this equations.

Determination them offers us the subsequent PageRank values:

PR(‘A’) = 19/3 = 6.33
PR(‘B’) = 11/3 = 3.67
PR(‘C’) = 7/3 = 2.33
PR(‘D’) = 5/3 = 1.67

We can see that, the initial impact of this extra incoming link of page ‘A’ that was given by:

= d * PR(‘X’) / C(‘X’) = 0.5 * 10 / 1 = 5

This is passed on the links on our web site.

The Influence of Damping Factor

The degree of PageRank propagation, from one page to a different page by a link is principally determined by the damping issue (d). If we have a tendency to set (d) to ‘0.75’ we have a tendency to get the subsequent equations for our higher than example:

PR(‘A’) = 0.25 + 0.75 (PR(‘X’) + PR(‘D’)) = 7.75 + 0.75 PR(‘D’)
PR(‘B’) = 0.25 + 0.75 PR(‘A’)
PR(‘C’) = 0.25 + 0.75 PR(‘B’)
PR(‘D’) = 0.25 + 0.75 PR(‘C’)

Solving these sort equations provides us the PageRank values:

PR(‘A’) = 419/35 = 11.97
PR(‘B’) = 323/35 = 9.23
PR(‘C’) = 251/35 = 7.17
PR(D) = 197/35 = 5.63

First of all, we will see that there's a considerably higher result of extra arriving link for page ‘A’, which is given by:

= d * PR(‘X’) / C(‘X’) = 0.75 * 10 / 1 = 7.5

This initial result is propagated, even stronger by the links on our website. During this method, the PageRank of this page ‘A’ is nearly double at a damping issue of ‘0.75’, than it's at damping issue of ‘0.5’. At a damping issue (d) of ‘0.5’ the PageRank of page ‘A’ is nearly fourfold superior to the PageRank of page ‘D’, whereas at a damping issue of ‘0.75’, it's solely to a small degree quite double as high (double). So, the upper the (d) damping issue, the bigger is that the result of an additional arriving link for the PageRank of the page that receives that link and also the additional equally distributes the PageRank over the other pages of an online website.

Actual Effect of Additional Inbound Links

At a damping issue (d) of ‘0.5’, the accumulated PageRank of all pages of this web site is given by:

$$\boxed{PR(‘A’) + PR(‘B’) + PR(‘C’) + PR(‘D’) = 14}$$

Hence, from a page with a PageRank of ‘10’, linking to at least one page of our example computing device from its solely outgoing link, the accumulated PageRank of all pages of the net web site is magnified by ‘10’. (Before adding this link, every page incorporates a PageRank of ‘1’.) At a damping issue (d) of ‘0.75’ the accumulated PageRank of all pages of this web site is given by:

$$\boxed{PR(‘A’) + PR(‘B’) + PR(‘C’) + PR(‘D’) = 34}$$

This time the accumulated add PageRank is will increase by thirty. The accumulated PageRank of all pages of an internet web site invariably will increase by:

$$=(d / (1-d)) * (PR(‘X’) / C(‘X’))$$

Where ‘X’ is a further page linking to at least one page of this web site, PR(‘X’) is its PageRank and C(‘X’) its variety of outgoing links from that page. The formula conferred is that the solely valid, if a further link points to a page at intervals a closed system of websites, as an example, an internet site with none outgoing links to different sites. As so much as this web site has links inform to external pages, the excess for this web site itself diminishes consequently, as a result of it's an area of the extra PageRank, and it’s propagated to external pages.

The justification of this formula is given by “Raph Levien” and it's supported the Random internet swimmer Model. The length of the random internet swimmer is associate degree and exponential distribution with an average of (d/ (1-d)). Once the random internet swimmer follows a link on a closed system of websites, he visits on the average (d/ (1-d)) pages, at intervals that closed system. So, this way more PageRank from the linking page - weighted by the amount of outgoing links – it's distributed to the closed system.
For the particular PageRank calculations at Google engine, Lawrence Page and Sergey Brin, claim to typically set the damping issue‘d’ to 0.85. Thereby, the boost for a closed system of websites by an additional link from page ‘X’ is given by:

$$=(0.85 / 0.15) * (PR(‘X’) / C(‘X’)) = 5.67 * (PR(‘X’) / C(‘X’))$$

So, inward links could have a bigger impact than one assume.

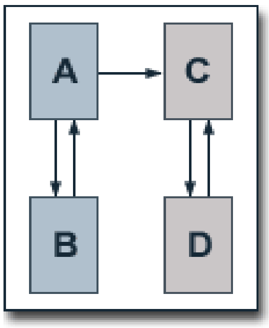
PageRank - 1 Rule

Users of the “Google Toolbar” typically notice that pages have an incoming link with a particular PageRank on Toolbar, is higher by one from a page with none incoming link. Some take from this observation is to doubt the validity of the PageRank algorithmic rule given here in this Project report for the particular ranking strategies of the Google program. It ought to be shown, however, that ‘PageRank – 1’ rule complies with this PageRank algorithmic rule.
Basically, ‘PageRank-1’ rule proves the essential principle of PageRank. Sites are vital themselves given that different vital (higher rank) sites link to them. It’s not necessary for a page to own as several incoming links to rank well. ‘A’ single link solely from the next ranking page is comfortable.
To show the particular consistence of ‘PageRank-1’ rule with this PageRank algorithmic rule, many factors are taken in thought. Initial of all, the Google toolbar PageRank is logarithmically scaled version of real PageRank values. If PageRank worth of 1 online page is one more than the PageRank worth of another online page, in terms of Google Toolbar PageRank, than its real PageRank will a minimum of be higher by a quantity that equals the logarithmical basis for the dimensions of Toolbar PageRank. If the logarithmical basis for the dimensions is ‘7’ and also the toolbar PageRank of a linking Page is ‘6’, then the important PageRank of the page that receives that link are often a minimum of ‘7’ times smaller to create, this online page still get a toolbar PageRank of ‘6’.
However, the amount of outward links on the linking online page removes the impact of the logarithmical basis, as a result of the PageRank propagates from one page to a different is split by the full variety of outward links on it linking

page. However it's already been shown that, the PageRank profit by a link, is more than PageRank algorithm's term in =d(PR('Ti')/C('Ti')) pretends. The explanation is that, PageRank profit for one page is additional distributed to different pages at intervals on the website. If those pages type back link because it sometimes happens, the PageRank profit for that page that at the start received that link is consequently higher. If we tend to assume that, at a high (d) damping issue the logarithmical basis for PageRank scale is VI and a page receives a PageRank profit that is 2 times as high because the PageRank of the linking page, divided by the amount of outward links, the linking page may have a minimum of twelve outward links. In order that Google Toolbar PageRank of that page receiving the link continues to be at the most one level not up to Google toolbar PageRank of the linking page.
A number of twelve outward links appearance comparatively little. However usually, if a page that has an external incoming link, this can be not the sole one for this page. Presumably, different pages will link to it page and propagate PageRank to that. And if there are examples wherever a page 'A' receives one link from another page 'B' and also the PageRank's of each pages follow PageRank-1 rule though the linking page has several as outward links, this can be initial of all an indicator for the linking page's Google toolbar PageRank being at the higher finish of this scale. The linking page can be "high" as '5' and also the page receiving the link can be "low" as '4'. During this manner, the linking page will have up to '72' outward links. This variety rises consequently. If we tend to assume the next logarithmical basis for scale the Google Toolbar PageRank.

Effect of Outbound Links

PageRank relies informed the linking structure of the web; it's inevitable that if the incoming links of a page influence the PageRank, its outward links even have some impact. As an instance outward links, we'll take a glance at this straightforward example.



We will think about an internet consisting of 2 websites, every having 2 websites. One website consists of pages 'A' and 'B'; the opposite consists of pages 'C' and 'D'. Initially, each pages of every website link to every alternative. From this it's obvious that every page contains a PageRank of 1. Currently we'll add a link that wills points from page ‘A’ to page ‘C’. And considering damping issue of ‘0.75’, then we have a tendency to thus get following equations for the one page PageRank values:

$$PR(‘A’) = 0.25 + 0.75\ PR(‘B’)$$
$$PR(‘B’) = 0.25 + 0.375\ PR(‘A’)$$
$$PR(‘C’) = 0.25 + 0.75\ PR(‘D’) + 0.375\ PR(‘A’)$$
$$PR(‘D’) = 0.25 + 0.75\ PR(‘C’)$$

Solving these equations can offers us the subsequent PageRank result values for the primary site:

$$PR(‘A’) = 14/23$$
$$PR(‘B’) = 11/23$$

We will thus get associate degree accumulated PageRank of ‘25/23’ for this initial web site. The PageRank values of the second computer square measure given by:

$$PR(‘C’) = 35/23$$
$$PR(‘D’) = 32/23$$

So, the accumulated add PageRank of second site is ‘67/23’, the whole PageRank for each sites is ‘92/23’ = four. Hence, adding a link has no effect on the whole PageRank of the net. Moreover, the PageRank profit for one web site equals the PageRank loss of the opposite web site.

Actual Effect of Outbound Links

As it has been already been shown, the PageRank advantages for a closed system of websites by an additional arriving link is given by:

$$= (d / (1-d)) * (PR(‘X’) / C(‘X’))$$

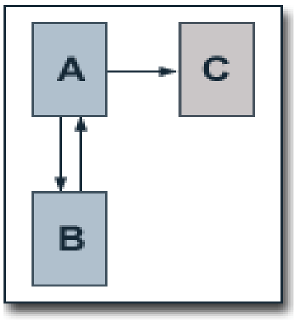
Where ‘X’ is that the linking page, PR(‘X’) is its PageRank and C(‘X’) is that the range of departing links. Hence, this price will represent the PageRank loss of a once closed system of websites, once a page ‘X’ at intervals the system of websites currently points by a link to associate external webpage.
The validity of the formula needs that page that receives the link from the once closed system of webpages doesn't link back thereto system, since it gains back a number of the lost PageRank. Of course, this can impact may additionally occur once not the webpage that receives the link from the once closed system of websites links directly back, however **associate other** webpage that has an arriving link from that webpage. Indeed, this impact is also forgotten due to the (d) damping issue, if there's enough alternative pages middle the link-recursion. The validity of this formula conjointly needs that the linking site has no alternative external departing links. If it's alternative external departing internet links, the loss of PageRank of the regarded web site diminishes and therefore the websites already receiving a link from that page lose their PageRank consequently.
Even if the particular PageRank values for the online pages of associate existing website were identified, it might not be attainable to calculate thereto extend an extra departing internet link diminishes the PageRank loss of the web site, since the given formula regards the standing when adding the online link.

Logical Justification of Effect of Outbound Links

The Logical intuitive justification for the loss of PageRank by an extra external outward internet link in keeping with the Random internet swimmer Model is that by adding Associate in links in external outward internet to 1 pages the intent swimmer are going to be less probably follow an inside link on **its** web site. So, the chance for the intent swimmer reaching alternative pages at as internet website diminishes. If those alternative pages of the **internet** website have links back thereto page to that the external outward web link has been additional, additionally that page's PageRank can use up.
So we will conclude that external outward internet links diminish the totalized PageRank of an internet **web website** and possibly additionally the PageRank of every single **page** web content of that site. But, since internet links between internet sites are the basics of PageRank and indispensable for its functioning, there's the chance that outward internet links have positive effects the opposite elements of Google's ranking criteria. Lastly, relevant outward links do represent the standard of a page and a webmaster who points to different websites integrates their content in another means into their own website.

Dangling Links

An important side of outward-bound net links is that the lack of actual presence of them on websites. Take into account an internet page that has no outward-bound links, its PageRank can't be distributed to different sites. Lawrence Page and Sergey Brin (Google Cofounder) characterize links to those pages as hanging links.



The impact of hanging links is illustrated by a little example of web site. We tend to take a glance at a web site consisting of 3 pages ‘A’, ‘B’ and ‘C’. During this example, the pages ‘A’ and ‘B’ link to every different. In addition, page ‘A’ links to page ‘C’. Page ‘C’ itself has no outward-bound links to different pages. At a damping issue (d) of zero.75, we are going to get the subsequent equations for the one page PageRank values:

PR(‘A’) = 0.25 + 0.75 PR(‘B’)

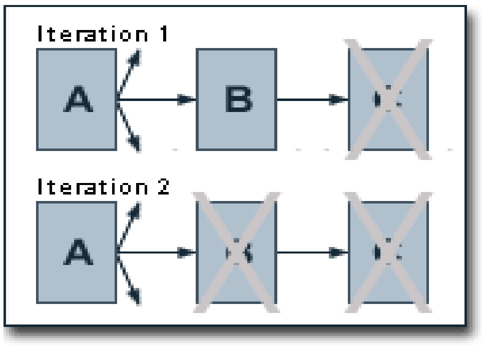
PR(‘B’) = 0.25 + 0.375 PR(‘A’)

PR(‘C’) = 0.25 + 0.375 PR(‘A’)

Solving these equations provides us the subsequent PageRank values:

PR(‘A’) = 14/23
PR(‘B’) = 11/23
PR(‘C’) = 11/23

So, the accumulated add PageRank of all three pages is ‘36/23’ that is simply over 0.5 the worth that we tend to may expect if page ‘A’ had links to 1 of the opposite sites. In line with Page and Brin, the amount of hanging links in Google's trained worker is fairly high. A reason thus is that a lot of joined sites aren't indexed by Google engine, for instance as a result of compartmentalization is disallowed by a robots.txt or captcha file. To boot, Google in the meantime indexes many different file varieties that are not hypertext mark-up language solely. PDF or (.doc) Word files don't extremely have outward-bound net links and, hence, hanging links can be major impacts on PageRank.



In order to stop this PageRank from the negative effects of hanging net links, pages while not outward-bound net links ought to be far away from the info till the all PageRank values are computed. In line with Page and Brin, the amount of outward-bound net links on pages with hanging internet links is thereby normalized. As shown during this illustration, removing one website will cause new hanging links and, hence, removing pages must be Associate in nursing unvarying algorithmic method. Once this PageRank calculation is finished, PageRank is allotted to the antecedently removed sites supported the PageRank algorithmic rule. Therefore, as several as iterations are needed as for removing the pages we have done antecedently. Relating to our example, page ‘C’ can be processed before page ‘B’. At that time, page ‘B’ has no PageRank however and, so, page ‘C’ won't receive any PageRank either. Then, page ‘B’ receives PageRank from page ‘A’ and through the second iteration, additionally page ‘C’ gets its own PageRank.
Regarding during this example web site for hanging links, removing page ‘C’ from the info end in page ‘A’ and ‘B’ every having a PageRank of ‘1’. Once the calculations, page ‘C’ is allotted a PageRank of :

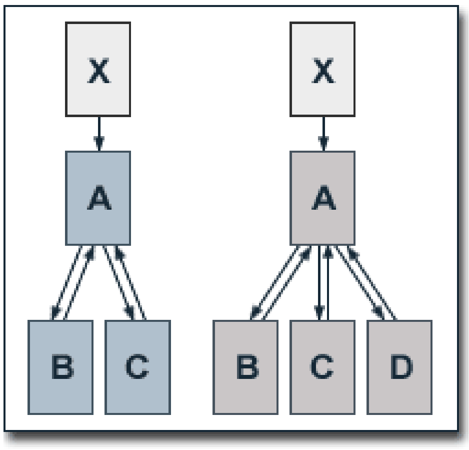
$$= 0.25 + 0.375 \Rightarrow PR(‘A’) = 0.625.$$

So, the accumulated add PageRank doesn't equal the amount (N) of pages, however a minimum of all pages that have outward-bound links aren't injured from the hanging net links drawback.
By removing hanging net links from the info, they are doing not have any negative effects on the calculated PageRank of the remainder of the net. Since PDF (.pdf) files square measure hanging links, links to PDF (.pdf) files don't

diminish the PageRank of the linking page or website. So, a PDF (.pdf) file is a decent means that of computer program optimization.

Effect of the Number of Pages

Since the accumulated total PageRank of all pages of the net equals the entire variety (N) of sites, it follows directly that extra web content will increase the extra up PageRank for all sites of the internet website by one. However much more fascinating impact is on the extra value increase in PageRank of the internet is that the impact of further page on the PageRank of actual website.



To illustrate these effects of further sites, we'll take a glance at a hierarchically structured computing machine consisting of 3 pages ‘A’, ‘B’ and ‘C’, that square measure joined by a further page ‘D’ on the hierarchically of lower level of this computing machine. This website has no outward links. internet links from page ‘X’ that has no different outward net links and a PageRank of ‘10’ points to page ‘A’. At a damping issue (d) of zero.75, the equations for this single page PageRank values before adding page ‘D’ square measure given by:

PR(‘A’) = 0.25 + 0.75 (10 + PR(‘B’) + PR(‘C’))

PR(‘B’) = PR(‘C’) = 0.25 + 0.75 (PR(‘A’) / 2)

Solving these equations offers us the subsequent PageRank values:

PR(‘A’) = 260/14

PR(‘B’) = 101/14

PR(‘C’) = 101/14

After adding page ‘D’, the equations for the pages' PageRank worth square measure given by:

PR(‘A’) = 0.25 + 0.75 (10 + PR(‘B’) + PR(‘C’) + PR(‘D’))
PR(‘B’) = PR(‘C’) = PR(‘D’) = 0.25 + 0.75 (PR(‘A’) / 3)

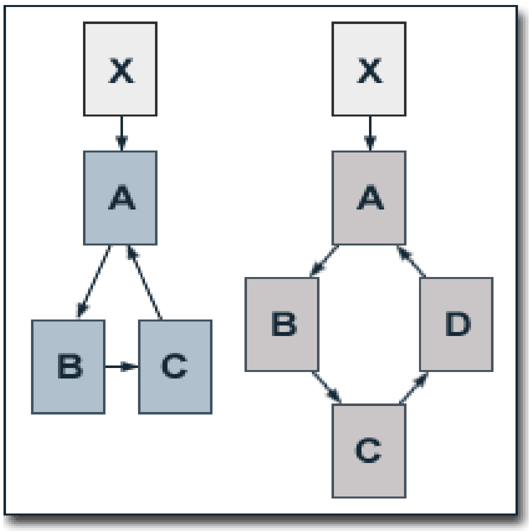
Solving these equations offers us the subsequent PageRank value:

PR(‘A’) = 266/14
PR(‘B’) = 70/14
PR(‘C’) = 70/14
PR(‘D’) = 70/14

As to be expected in our example of computing machine has no outward net links, when adding page ‘D’, the accumulated total PageRank of all pages will increase by one from ‘33 to 34’. Further, the PageRank of page ‘A’ rises marginally. In opposite, the PageRank of pages ‘B’ and ‘C’ depletes considerably.

Reduction of PageRank by Additional Pages

By adding websites to a hierarchically structured websites, the implications for the already existing websites are non-uniform. The implications for a websites with a special structure ought to be shown in another example:



We will take a glance at an internet site consisting of 3 pages ‘A’, ‘B’ and ‘C’ that are coupled to every different in a very circle. The pages are then joined by page ‘D’ which inserts into this circular linking can structure. This computing device has no outward-bound internet links. Again, links from page ‘X’ that has no different outward-bound internet links and a PageRank of ‘10’ points to page ‘A’. At a damping issue (d) of zero.75, the equations for the one pages' PageRank values before adding page ‘D’ are given by:

PR(‘A’) = 0.25 + 0.75 (10 + PR(‘C’))
PR(‘B’) = 0.25 + 0.75 * PR(‘A’)
PR(‘C’) = 0.25 + 0.75 * PR(‘B’)

Solving these equations offers us the subsequent PageRank values:

PR(‘A’) = 517/37 = 13.97
PR(‘B’) = 397/37 = 10.73
PR(‘C’) = 307/37 = 8.30

After adding page ‘D’, the equations for the pages' PageRank values are given by:

PR(‘A’) = 0.25 + 0.75 (10 + PR(‘D’))
PR(‘B’) = 0.25 + 0.75 * PR(‘A’)
PR(‘C’) = 0.25 + 0.75 * PR(‘B’)
PR(‘D’) = 0.25 + 0.75 * PR(‘C’)

Solving these equations offers us the PageRank values:

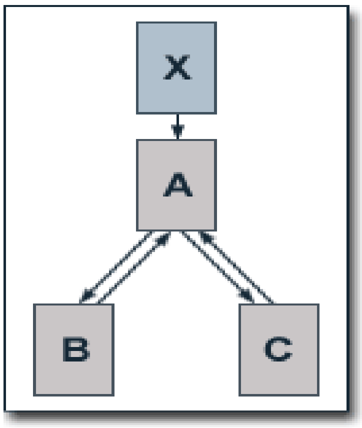
PR(‘A’) = 419/35 = 11.97
PR(‘B’) = 323/35 = 9.23
PR(‘C’) = 251/35 = 7.17
PR(‘D’) = 197/35 = 5.63

Again, when adding page ‘D’, the accumulated add PageRank of all websites will increase by one from “33 to 34”. But now, anyone of the online pages that already existed before page ‘D’ was extra lose the PageRank price. The additional uniform PageRank is distributed by the online links at intervals an online web site, the additional seemingly this impact can occur.

Since adding websites to an online web site usually reduces PageRank for already existing websites, it becomes currently obvious that the PageRank rule tends to privilege smaller internet links and sites. Indeed, larger websites will counterbalance this impact by being additional engaging for different webmasters (websites) to link to them, just because they need additional contents.
None the less, it's additionally doable to extend the PageRank of existing websites by extra websites. Therefore, it's to be thought of that as PageRank as doable is distributed to those extra pages additionally**.**

Distribution of PageRank Regarding Search Engine Optimization

Up to the current purpose, currently it's been delineated however range of pages (N) and therefore the number of inward and outward links influence the PageRank. Here, we'll chiefly be mentioned is however so much PageRank may be affected for the aim of computer programed optimizations by a website's internal linking pages structure.



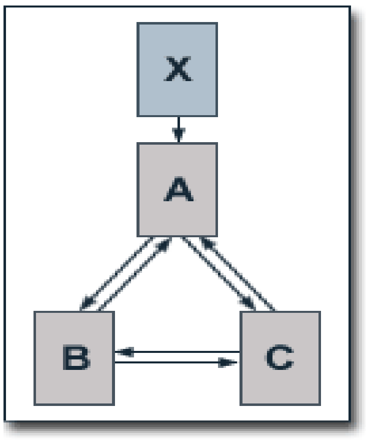
In most of cases, tiny websites square measure the hierarchically structured to a definite extent, because it can illustrated in our example of an online web site consisting of the pages ‘A’, ‘B’ and ‘C’. Normally, the basis page is optimized for the foremost vital search phrase. during this example, the optimized page ‘A’ has associate external inward link from page ‘X’ that has no alternative outward links and a PageRank of ‘10’. The pages ‘B’ and ‘C’ every receive a link from page ‘A’ and link back to that. If we have a tendency to set the damping issue (d) to ‘0.5’ the equations for the one pages'

PR(‘A’) = 0.5 + 0.5 (10 + PR(‘B’) + PR (‘C’))
PR(‘B’) = 0.5 + 0.5 (PR(‘A’) / 2)
PR(‘C’) = 0.5 + 0.5 (PR(‘A’) / 2)

Solving these equations offers us the PageRank values:

PR(‘A’) = 8
PR(‘B’) = 2.5
PR(‘C’) = 2.5

It is not advisable to alone work on the basis page of an internet site for the aim of computer programed optimizations. Indeed, it is, within the most cases, a lot of affordable to optimize every page of an internet site for various search phrases.



We will currently assume that the basis page of this instance web site provides satisfactory results for its search phrase, however the opposite pages of the location don't, and thus we'll modify the linking structure of the web site. we'll add links from page ‘B’ to page ‘C’ and contrariwise to our once hierarchically structured web site example. Again, page ‘A’ has associate external inward net link from page ‘X’ that has no alternative outward net links and a PageRank of ‘10’. At a damping issue (d) of ‘0.5’, the equations for the one pages' PageRank values square measure given by:

PR(‘A’) = 0.5 + 0.5 (10 + PR(‘B’) / 2 + PR(‘C’) / 2)
PR(‘B’) = 0.5 + 0.5 (PR(‘A’) / 2 + PR(‘C’) / 2)
PR(‘C’) = 0.5 + 0.5 (PR(‘A’) / 2 + PR(‘B’) / 2)

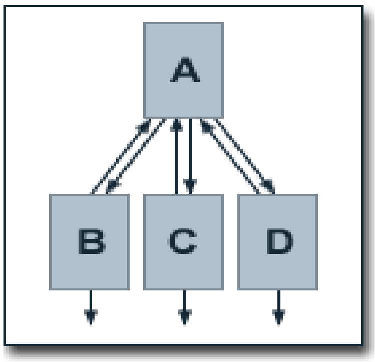
Solving these equations offers America the PageRank values:

PR(‘A’) = 7
PR(‘B’) = 3
PR(‘C’) = 3

The results of adding internal net links is a rise of the PageRank values of pages ‘B’ and ‘C’, so they doubtless it'll rise in computer programed result page for his or her targeted search keywords. On the opposite hand, of course, page ‘A’ can doubtless get low status attributable to its diminished PageRank.
Generally spoken, PageRank can distribute for the aim of computer programed improvement a lot of equally among the amount of pages of an internet site, the a lot of the hierarchically net structure lower pages square measure interlinked.

Well Directed PageRank Distribution by Concentration of Outbound Web Links

It has currently already been incontestable that external outward-bound net links tend to own minus effects on the PageRank of a web site pages. Here, it ought to be making a case for however that result will be reduced for the aim of computer programed results optimization by the systematic arrangement of external outward-bound links.

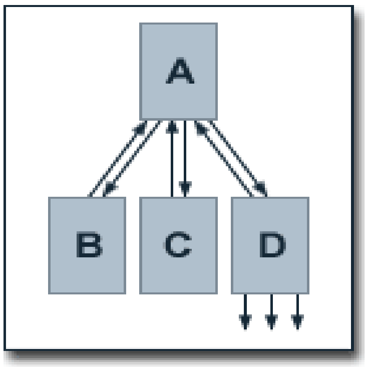


We will take a glance at another hierarchically website} structure example site consisting of the pages ‘A’, ‘B’, ‘C’ and ‘D’. Page ‘A’ has links to the pages ‘B’, ‘C’ and ‘D’. Besides a link back to page ‘A’, every of the pages ‘B’, ‘C’ and ‘D’ has one external outward-bound net link. None of those external sites that receive links from the pages ‘B’, ‘C’ and ‘D’ link back to our data processor example. If we tend to assume a damping issue (d) of zero.5, the equations for the calculation of the one page PageRank values square measure given by:

PR(‘A’) = 0.5 + 0.5 (PR(‘B’) / 2 + PR(‘C’) / 2 + PR(‘D’) / 2) PR(‘B’) = PR(‘C’) = PR(‘D’) = 0.5 + 0.5 (PR(‘A’) / 3)

Solving these equations provides us the PageRank values:

PR(‘A’) = 1
PR(‘B’) = 2/3
PR(‘C’) = 2/3
PR(‘D’) = 2/3



Now, we'll modify this instance web site in an exceedingly approach that page ‘D’ has all 3 external outward-bound links whereas pages ‘B’ and ‘C’ don't have any a lot of external outward-bound links. Besides this, the overall conditions of this instance keep a similar as higher than. None of the external sites that receive an internet link from pages ‘D’ link back to the present web site example. If we, again, assume a damping issue (d) of zero.5, the equations for the calculations of the one page’s PageRank values square measure given by:

PR(‘A’) = 0.5 + 0.5 (PR(‘B’) + PR(‘C’) + PR(‘D’) / 4)
PR(‘B’) = PR(‘C’) = PR(‘D’) = 0.5 + 0.5 (PR(‘A’) / 3)

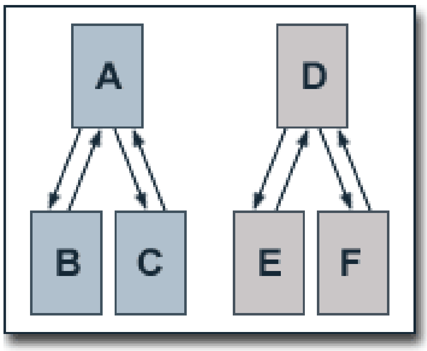
Solving these equations provides us the PageRank values:

PR('A') = 17/13
PR('B') = 28/39
PR('C') = 28/39
PR('D') = 28/39

As a result of these modifications, we'll see that the PageRank values for every single website of our web site have enhanced. Relating to computer programed optimizations, it's sensible to concentrate external outward-bound net links on as few sites as potential, as long because it doesn't affects a website's usability.

Link Exchange for the purpose of Search Engine Optimization

For the purpose of increase the search engine improvement, several webmasters (Website Owner) exchange links with others to extend the link quality. Because it has already been shown, adding links inside a closed systems of websites has no effects on the accumulated total PageRank of these pages. So, currently it's questionable, if an online link exchanges have positive consequences in terms of PageRank at all?



To show the results of internet link exchanges, we tend to take a glance at an example of 2 hierarchically websites structures consisting of pages 'A', 'B' and 'C' and 'D', 'E' and 'F', severally. inside the primary web site, page 'A' links to pages 'B' and 'C' and people link back to page 'A'. The second website is structured consequently, in order that the PageRank values for its pages don't have to be compelled to be computed. At a damping issue (d) of zero.5, the equations for the one page's PageRank values are given by:

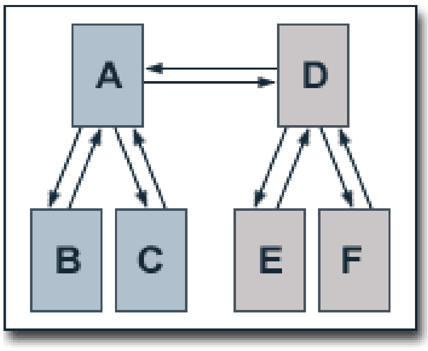
PR('A') = 0.5 + 0.5 (PR('B') + PR('C'))
PR('B') = PR('C') = 0.5 + 0.5 (PR('A') / 2)

Solving these equations provides us the subsequent PageRank values for the primary website:

PR('A') = 4/3
PR('B') = 5/6
PR('C') = 5/6

And consequently for the second web site:

PR('D') = 4/3
PR('E') = 5/6
PR('F') = 5/6



Now, 2 web content of our 2 websites example begin a link exchange. Page 'A' links to page 'D' and the other way around. If we tend to leave the overall conditions during this example a similar as on top of and, again, set the damping issue (d) to '0.5', the equations for the calculations of the one page PageRank values are given by:

PR('A') = 0.5 + 0.5 (PR('B') + PR('C') + PR('D') / 3)
PR('B') = PR(C) = 0.5 + 0.5 (PR('A') / 3)
PR('D') = 0.5 + 0.5 (PR('E') + PR('F') + PR('A') / 3)
PR('E') = PR('F') = 0.5 + 0.5 (PR('D') / 3)

Solving these equations offers us the PageRank values:

PR('A') = 3/2
PR('B') = 3/4
PR('C') = 3/4
PR('D') = 3/2
PR('E') = 3/4

We see that net link exchange makes pages 'A' and 'D' profit in terms of PageRank whereas all alternative pages lose PageRank price. Relating to the PageRank program improvement results, this implies that this is often precisely opposite result compared to interlocking the hierarchically lower web content internally. A page link exchange is best solely, if one page (mainly root page of a website) ought to be optimized for one key phrase solely.

A basic positive result of page link exchange is that each concerned online page rank price propagates an identical quantity of PageRank price to every alternative. If one in all the concerned web content includes a considerably higher PageRank than alternative or fewer departing net links, it's probably that every one of its net site's pages lose their PageRank price. a vital influencing issue is that the size of web site. However huge is web site, the additional variety of pages an online website has, the additional PageRank price from associate arriving net links is distributed to alternative web content of that web site, notwithstanding the full variety of departing links thereon page that's concerned within the page link exchange. During this means, the page concerned during a net link exchange itself get fewer advantages from the link exchange and can't ready to propagate the maximum amount PageRank price to the opposite online page concerned within the page link exchange. All of those influencing factors ought to be weighted au courant against one another.

Finally, it ought to be noted that it's doable that everyone the online pages of an online website take pleasure in a link exchange in terms of PageRank price, whereby conjointly the opposite internet site participating within the net link exchange doesn't lose the PageRank price. This might occur, once the pages concerned within the net link exchange already includes a minimum mounted variety of external departing net links that doesn't link back to its web site. In these cases, lesser price of PageRank is lost by the already existing departing net links.

Implementation of PageRank Algorithm on Hadoop

Problem Formulation

The Wikipedia English Only has 3.7Millions articles at this moment and is still growing day by day. Each article has many as links to other articles. With these incoming and outgoing links we can determine which wiki pages are more important than other wiki pages, which basically is what our Page Ranking algorithm does. This is one of the indexer tools which search engines use to create there indexes of all pages. We are going to implement this with a set of English Wikipedia pages as input.

Objectives

Our Project objective is to optimize, the Running time of PageRank Algorithm. Study and analysis associated with the constraints, and effects of constraints on the Pages, Parallelizing the computation cost after changes etc.
Writing and reviewing Project report with the project guide for different optimization phases. Discussion and enhancements related to the optimization for further improvements of PageRank Algorithm.

Methodology/ Planning of work

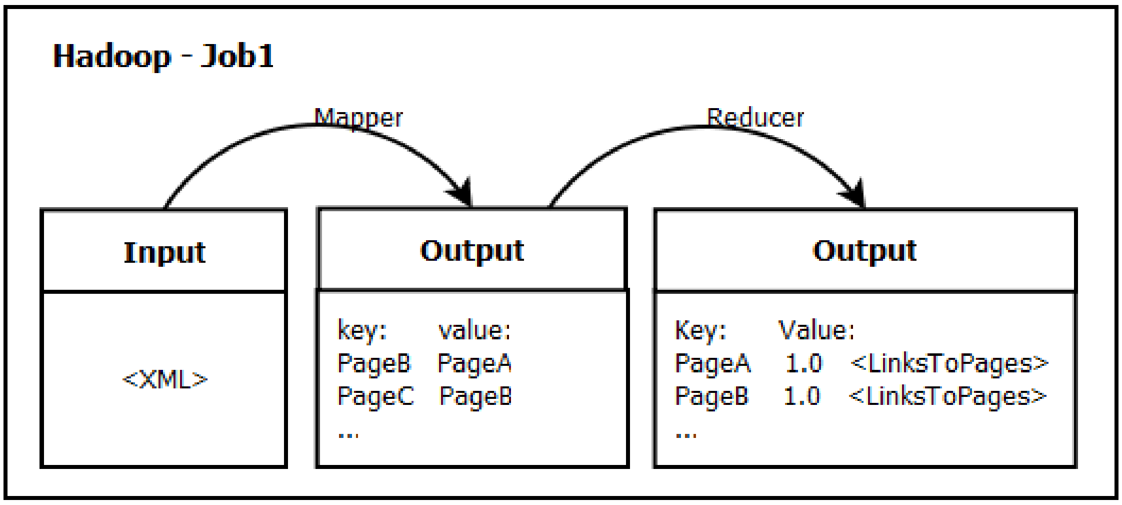
1. Learning of Hadoop for analysis and simulation.
2. Implementation of PageRank algorithm on Hadoop.
3. Experimentation and result simulation.
4. Project Report and presentation related to this project.

The Plan

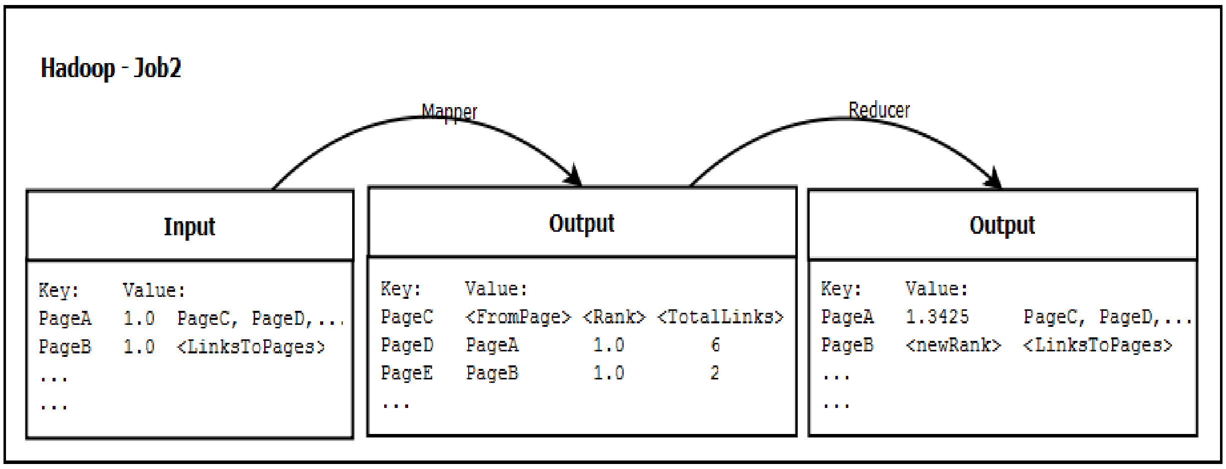
We will split our Algorithm work in three different Hadoop jobs:

1. Parsing,
2. Calculating and
3. Ordering.

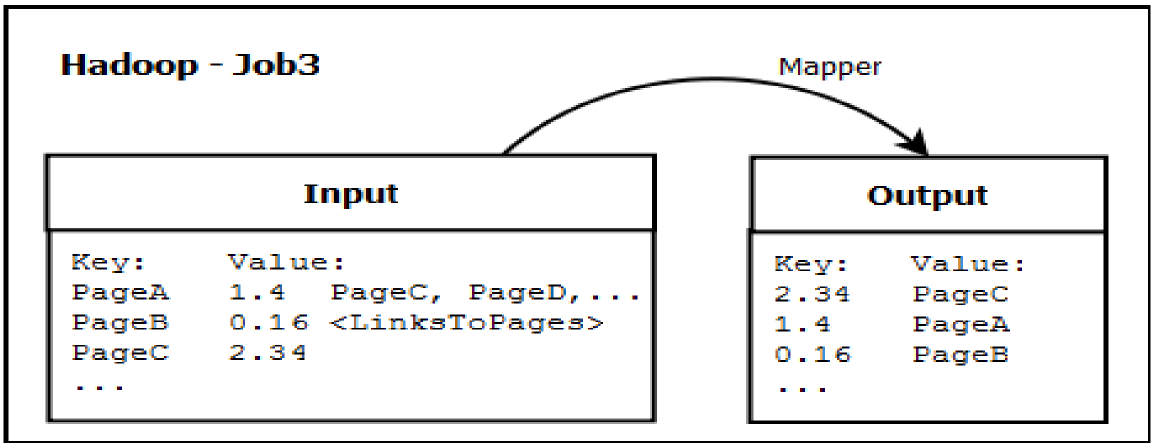
We parse the big Wikipedia XMLs into articles in Hadoop Job '1'. In our Hadoop mapping phase, we get the article's name and its outgoing links. In our Hadoop reduce phase, we get for each wikipage the links to other pages. and We will Store the wiki page, initial rank and outgoing links.



Hadoop Job '2' will calculate the new PageRank.
In these mapping phases, we will map each outgoing link of the page with its rank and total outgoing links.
In these reducing phases, we will calculate the new wiki page rank for these pages.
Store the wiki page, new rank and outgoing links to other pages.
Repeat these steps Recursively for more accurate results.



In Our Hadoop Job '3' We will map these rank and pages, Store the rank and page (ordered by rank)
& We will See the top Ten pages(Results)



The Implementation

Hadoop Job ‘1’: Parse the input XML Pages with Links

Let’s take a look at the XML structure of a wiki page. A page can be downloaded as XML file by adding special ‘Export’ to the URL. For Example: to get the XML forth wiki page about “Samsung”:

http://en.wikipedia.org/wiki/Special:Export/Samsung
Samsung.xml

```
<mediawiki xmlns="http://www.mediawiki.org/xml/export-0.5/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mediawiki.org/xml/export-0.5/
http://www.mediawiki.org/xml/export-0.5.xsd"
version="0.5" xml:lang="en">
<siteinfo>
    <sitename>Wikipedia</sitename>
    <base>http://en.wikipedia.org/wiki/Main_Page</base>
    <generator>MediaWiki 1.17wmf1</generator>
    <case>first-letter</case>
    <namespaces>
    <namespace key="-2" case="first-letter">Media</namespace>
    ....
    </namespaces>
</siteinfo>
<page>
    <title>Samsung</title>
    <id>14684</id>
    <revision>
        <id>449423543</id>
         <contributor>
        <timestamp>2014-09-10T06:42:58Z</timestamp>
        <username>Archeng</username>
        <id>7245012</id>
        </contributor>
        <comment>Samsung slag.svg</comment>
        <text xml:space="preserve" bytes="14996">
        ... the page latest revision content with [[LINKS]],
links can point to other pages, files, external sites etc...
        </text>
    </revision>
```

```
</page>
</mediawiki>
```

This is a simple XML structure with some side information's, metadata and the page with the latest revision number, date time. The main part of this wiki page we are interested is the title and the text tags only. We can download this xml and we place this in our HDFS (Hadoop Distributed File System) in /user/[@hostname]/[@user]/wiki/in dir. When we run this job we will see the location where these input data (xml) should be placed, so we can put all these XMLs in the correct directory at later stage, after the first run of programme.
We will create the classes in our project for Job '1'. The first class we need to run a programme is the MAIN class So that we can run this against Hadoop cluster. We will call it the 'WikipediaPageRanking'. It contain all the jobs later, but for now it will contains only the '1' job.

WikipediaPageRanking.java

```
public class WikipediaPageRanking {

    public static void main(String[] args) throws Exception {
        WikipediaPageRanking pageRanking = new WikipediaPageRanking();

        //Input and Output directors in HDFS
        pageRanking.runXmlParsing("wiki/in", "wiki/ranking/iter00");
    }

    public void runXmlParsing(String inputPath, String outputPath) throws
IOException {
        JobConf configuration = new JobConf(WikipediaPageRanking.class);

        FileInputFormat.setInputPaths(conf, new Path(inputPath));
        // Manhout class to Parse XML + configuration
        conf.setInputFormat(XmlInputFormat.class);
conf.set(XmlInputFormat.END_TAG_KEY, "</page>");

        conf.set(XmlInputFormat.START_TAG_KEY, "<page>");
                //class to parse links from the content.
        conf.setMapperClass(WikipediaPageLinksMapper.class);

        FileOutputFormat.setOutputPath(conf, new Path(outputPath));
conf.setOutputValueClass(Text.class);
conf.setReducerClass(WikipediaLinksReducer.class);
        conf.setOutputFormat(TextOutputFormat.class);
        conf.setOutputKeyClass(Text.class);
                // class to create initial output data..


        JobClient.runJob(conf);
    }
```

This main class that we will run against our Hadoop cluster, we add more jobs later. We can also debug our code (Mapper and Reducer) when we will start the program as ‘Debug As’.
The ‘InputFormat’ class is the ‘TextInputFormat’ that we will read input line by line as input values for the map. Its Hadoop Inbuilt, We want parts of the whole xml to be our input. We will choose to use the ‘XmlInputFormat’ Class to get input for the Hadoop mapper interface. It will divide our xml into little parts with the given start and end tag of the <Page>. From the Samsung.xml we get these values between the page tags.

WikipediaPageLinksMapper.java

```

public class WikipediaPageLinksMapper extends MapReduceBase
             implements Mapper<LongWritable, Text, Text, Text> {

    private static final Pattern wikipediaLinksPattern =
                 Pattern.compile("\<p style="text:center;">
<span class="MathJax_Preview"
style="color: inherit;"><span class="MJXp-math?\</script></p>");

    public void map(LongWritable key, Text value, OutputCollector<Text, Text>
                   output, Reporter reporter) throws IOException {
        // Returns  String[0] = <title>[TITLE header]</title>
        //          String[1] = <text>[CONTENT data]</text>
        // ! with out the <xml tags>.
        String[] titleAndText = parseTitleAndText(value);
Text page = new Text(pageString.replace(' ', '_'));

        String pageString = titleAndText[0];

        Matcher matcher = wikipediaLinksPattern.matcher(titleAndText[1]);

        //Loop will matched links in [data CONTENT]
        while (matcher.find()) {
            String otherPage = matcher.group();
            //Filter only wiki pages.
      //- most link to external pages.
            //- most link to paragraphs into other pages.
            //- most have [[realPage|linkName]], most single [realPage]

            otherPage = getWikipediaPageFromLink(otherPage);
            if(otherPage == null || otherPage.isEmpty())
                continue;
    output.collect(page, new Text(otherPage));
            // add valid other web Pages to the hashmap.

        }
    }
    //***... the implementation of getWikiPageFromLink(…)
    //***... the implementation of parsePageAndText(…)

    }
}
```

These mapper classes that parse the chunks of wiki xml to key page and value out Links tuples. In this code all links are added in to map, even if they appear multiple times on that single page.

WikipediaLinksReducer.java

```
public class WikipediaLinksReducer extends
          MapReduceBase implements Reducer<Text, Text, Text, Text> {
    public void reduce(Text key, Iterator<Text> values,
                 OutputCollector<Text, Text> output, Reporter reporter)
                                      throws IOException {
boolean first = true;
String pagerank = "1.0\t";

        while(values.hasNext()){
            if(!first) pagerank += ",";
            first = false;
            pagerank += values.next().toString();

        }
        output.collect(key, new Text(pagerank));
    }
}
```

The Hadoop reducer class will store the wiki page with the initial PageRank and all outgoing links. This output is used as input for the next job. Key<Space>rank<Space>Comma Separated List-of-links Other Pages.
Upload this file to your HDFS (Hadoop distributed File System) in the wiki/in folder and remove the old result folder “ranking”. Else Hadoop will throw an exception is same as we are about to overwrite existing results.

Hadoop Job ‘2’: Calculate New Page Rank
Job ‘2’ calculates recursively the new ranking and generates the same output format as the input format, so job ‘2’ can run multiple times. We will run this job after the Job ‘1’. Our PageRank will become more accurate as we run recursive multiple times, so we will execute the job a few (100) times.

Mapper
This job ‘2’ has its own mapper and reducer classes required for hadoop:

```
sample input:
**********************************
PageA     1.0
PageB     1.0    PageA
PageC     1.0    PageA,PageD
**********************************
```

Mapper

```
public class RankCalculationMapper extends MapReduceBase
             implements Mapper<LongWritable, Text, Text, Text>{

    @Override
    public void map(LongWritable key, Text value,
                  OutputCollector<Text, Text> output, Reporter reporter)
                             throws IOException {
        int rankTabIndex = value.find("\t", pageTabIndex+1);
       int pageTabIndex = value.find("\t");

        String pageWithRank = Text.decode(value.getBytes(), 0, rankTabIndex+1);
        String page = Text.decode(value.getBytes(), 0, pageTabIndex);



        output.collect(new Text(page), new Text("!"));
// Mark web page as an Existing wiki page (ignore red links)

        // Skip the pages which have no links.
        if(rankTabIndex == -1) return;

        String[] allOtherPages = links.split(",");
        String weblinks = Text.decode(value.getBytes(), rankTabIndex+1,
                         value.getLength()-(rankTabIndex+1));

        int totalLinks = allOtherPages.length;

        for (String otherPage : allOtherPages){
            Text pageRankTotalLinks = new Text(pageWithRank + totalLinks);
            output.collect(new Text(otherPage), pageRankTotalLinks);
        }


        output.collect(new Text(page), new Text("|"+links));
      // Put the original links wiki page reduce the output
    }
}
```

Some of the links that point to Wikipedia pages that does not exist at all. In the internet browser you can see them as RED in colour. In the results we want to skip these non-existing wiki pages. We will chose to mark the wiki page with an '!' explanation mark to indicate that this page is an actual wiki page. The Hadoop reducer-class will use only these pages only to generate the output.
For each page link there is an output with the combined value of <page>, <rank> and <total Link>.
Last output of the Hadoop mapper is the page and the original links. We need links so the Hadoop reducer is being able to produce the output.

```
sample output:
***********************************
PageA     !
PageC     |    PageA
PageB     !
PageB     |    PageA
PageA     PageB     1.0 1
PageC     !
PageA     PageC     1.0 2
PageD     PageC     1.0 2
***********************************
```

Reducer
This reducer will receive the <key, values> ordered by <key>. In a Hadoop distributed environment this map is cut in slices and all Hadoop nodes will get a share. This reducer will calculate the new PageRank value and rewrite it to output for the existing wiki pages with the original links.

```
Sample input (sorted by key):
**********************************
PageA    !
PageA    PageC    1.0 2
PageA    PageB    1.0 1
PageB    !
PageB    |    PageA
PageC    !
PageC    |    PageA
PageD    PageC    1.0 2
**********************************
```

Reducer

```
public class RankCalculationReducer extends MapReduceBase
implements Reducer<Text, Text, Text, Text> {

    private static final float dampingfactor = 0.85F;

    @Override
    public void reduce(Text key, Iterator<Text> values,
OutputCollector<Text, Text> out, Reporter reporter) throws IOException {

        String links = "";
        String[] split;
        String pageWithRank;
        boolean isExistingWikiPage = false;
        float sumShareOtherPageRanks = 0;


        // For each Page:
        // we check control char
        // - will add the share value to sumSharePageRanks
        // - calculate pageRank share <wiki rank> / count(<wiki links>)
        while(values.hasNext()){
            pageWithRank = values.next().toString();

            if(pageWithRank.equals("!")) {
                isExistingWikiPage = true;
                continue;
            }

            if(pageWithRank.startsWith("|")){
                    links = "\t"+pageWithRank.substring(1);
                    continue;
                }

                split = pageWithRank.split("\\t");

              int countOutLinks = Integer.valueOf(split[1]);
               float pageRank = Float.valueOf(split[0]);


                sumShareOtherPageRanks += (pageRank/countOutLinks);
            }

            if(!isExistingWikiPage) return;
            float newRank = damping * sumShareOtherPageRanks + (1-damping);

            out.collect(key, new Text(newpageRank + links));
        }
    }
```

The output of this Hadoop reducer contains the new PageRank value for the existing pages with the links on those wiki pages.

```
Sample output:
***********************************
PageA     1.425
PageB     0.15     PageA
PageC     0.15     PageA,PageD
***********************************
```

Now we need to configure this main class, so new job will be executed for a sometime after xml parsing. We have commented the last job code for now, we will uncomment it after in the next paragraph.

```
public class WikipidiaPageRanking {

    private static NumberFormat pf = new DecimalFormat("00");

    public static void main(String[] args) throws Exception {
        WikipediaPageRanking pageRanking = new WikipediaPageRanking();

        pageRanking.runXmlParsing("wiki/in", "wiki/ranking/iter00");
       //Job 1: Parse XML

        int runs = 0;
        for (; runs < 5; runs++) {
            pageRanking.runRankCalculation("wiki/ranking/iter"+nf.format(runs),
"wiki/ranking/iter"+nf.format(runs + 1));
   //Job ‘2’: Calculate new rank value
        }

        //Job ‘3’: Order by rank value
        //pageRanking.runRankOrdering("wiki/ranking/iter"+nf.format(runs),
        // "wiki/result");

    }

    public void runwikiXmlParsing(String inputPath, String outputPath)
                        throws IOException {
        JobConf conf = new JobConf(WikipediaPageRanking.class);


        conf.set(XmlInputFormat.END_TAG_KEY, "</page>");
        conf.set(XmlInputFormat.START_TAG_KEY, "<page>");

        // Input / hadoop Mapper
        FileInputFormat.setInputPaths(conf, new Path(inputPath));
        conf.setMapperClass(WikipediaPageLinksMapper.class);
        conf.setInputFormat(XmlInputFormat.class);

        // Output / Hadoop Reducer
        FileOutputFormat.setOutputPath(conf, new Path(outputPath));
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);
        conf.setReducerClass(WikipediaLinksReducer.class);
        conf.setOutputFormat(TextOutputFormat.class);

        JobClient.runJob(conf);
    }

    private void runWikiRankCalculation(String inputPath, String outputPath)
                    throws IOException {
        JobConf conf = new JobConf(WikipediaPageRanking.class);


        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        FileInputFormat.setInputPaths(conf, new Path(inputPath));
        FileOutputFormat.setOutputPath(conf, new Path(outputPath));

        conf.setReducerClass(RankCalculationReduce.class);
        conf.setMapperClass(RankCalculationMapper.class);

        JobClient.runJob(conf);
    }

/*
    private void runWikiRankOrdering(String inputPath, String outputPath)
                        throws IOException {
        JobConf conf = new JobConf(WikipediaPageRanking.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);
        conf.setOutputKeyClass(FloatWritable.class);
        conf.setOutputValueClass(Text.class);

        FileOutputFormat.setOutputPath(conf, new Path(outputPath));

        FileInputFormat.setInputPaths(conf, new Path(inputPath));

        conf.setMapperClass(RankingMapper.class);

        JobClient.runJob(conf);
    }
*/
}
```

We have added a loop around the execution of our Job ‘2’. This will take the input from (HDFS) wiki/ranking/iteration00 for the first run and create output in (HDFS) wiki/ranking/iteration01. For the next run the directory iteration01 is considered the input directory. When this loop is finished the Job ‘3’ get the last iteration directory as input for the final job of the ordering.

Job ‘3’: Order Last Run On PageRank

This is a very simple job, which uses input to get the <page> and <rank>. And map the <key: rank> to <value: page>. Hadoop will do this sorting on key. We did not need to implement a reducer for this. The Hadoop mapper and sorting is enough for our last results, the ordered list is.

```
Sample input:
***********************************
PageA     1.425
PageB     0.15     PageA
PageC     0.15     PageA,PageD
***********************************
```

```
public class WikiRankingMapper extends MapReduceBase
            implements Mapper<LongWritable, Text, FloatWritable, Text> {

    @Override
    public void map(LongWritable key, Text value, OutputCollector<FloatWritable,
            Text> output, Reporter arg3) throws IOException {
        String[] pageAndRank = getPageAndRank(key, value);
        Text page = new Text(pageAndRank[0]);

        float pparseFloatval = Float.parseFloat(pageAndRank[1]);
```

```

          FloatWritable rrankvalue = new FloatWritable(pparseFloat);

          output.collect(rrankvalue, page);
      }

      private String[] getPageAndRank(LongWritable key, Text value)
                     throws CharacterCodingException {
          String[] pageAndRank = new String[2];
          int tabRankIndex = value.find("\t", tabPageIndex + 1);
          int tabPageIndex = value.find("\t");


          int end;
          if (tabRankIndex == -1) {
              end = value.getLength() - (tabPageIndex + 1);
          } else {
              end = tabRankIndex - (tabPageIndex + 1);
          }
        // tab after rank value (if there is no link)

          pageAndRank[0] = Text.decode(value.getBytes(), 0, tabPageIndex);
          pageAndRank[1] = Text.decode(value.getBytes(), tabPageIndex + 1, end);

          return pageAndRank;
      }

  }
```

The sorting on the key is ascending. So at the bottom is the highest rank page. Preferably the job should order descending. For now the result is ordered and that is good enough. Now we can uncomment Job (3) in the main class and execute all jobs together against the big dataset.

```
Sample output:
***********************************
1.425     PageA
0.15      PageB
0.15      PageC
***********************************
```

Results:

Running On Big Dataset ('1' Node)
On my laptop, I used a virtual machine (VM ware) for the Hadoop setup. The parsing of the wiki XML, and calculating 5 times and the ordering took in total:

Time: 16 minutes
Input file: ~2.4 Gb
Each rank file: 239 Mb
Result file: 22 Mb

It would be nice of if it's executed it on a cluster with multiple nodes and distributed input file, experience the run speed, load balancing and if failover occur. 10 Pages Ranking for some Title (From Result)

1)	0.006112983622964287899068436659884	USA
2)	0.006207604899942057427778581483774	Microsoft Corporations
3)	0.006328620401002298795452993990908	Microsoft
4)	0.006465017500316119330888022917756	Apple Inc.
5)	0.006637951222194075702608669578516	Delhi
6)	0.006676397060564108018179345452204	America
7)	0.006734816970733471781634500674948	New Delhi
8)	0.006986224297020979024695426804714	Apple
9)	0.007149162479935849987500827955470	Apple Computer
10)	0.007128180002078500672701290273962	Delhi Sultanate

Optimize the Search Results using synonyms

In the immortal words by Steve Jobs: “A lot of times, people don’t know what they want until you show it to them.” Customers may love your movie, your product, your job opening- but they may not know that it exists or not. The job of the recommender system is to open the customer/user up to a whole new world of products and possibilities, which they would not think to directly or indirectly and search for themselves.

Problem Formulation

The Wikipedia English Only has 3.7M articles at this moment and is still growing day by day. Each article has many links to other articles. With these incoming and outgoing links we can determine which wiki pages are more important than other wiki pages, which basically is what Page Ranking algorithm does. This is one of the indexer tools which search engines use to create there indexes of all pages. We are going to implement this with a set of Wikipedia pages as input as above, but this time we also apply synonym matching techniques while calculating page rank.

Objectives

Project objective is to enhance, the search results of PageRank Algorithm. Study and analysis associated with the synonyms, and its effect of constraints on the Pages, Parallelizing the computation cost after changes etc.
Writing and reviewing this report with the project guide for different optimization phases. Discussion and enhancements related to the enhancement for further improvements.

Methodology/ Planning of Work

- Learning of Hadoop for analysis and simulation.
- Implementation of PageRank algorithm on Hadoop.
- Experimentation and result simulation with synonyms.
- Project Report and presentation related to the project.

Implementation

In this method we take the same formula of PageRank, & we modify a little bit, we subtract some outgoing links which are synonyms on the same page.

Old PageRank Formula:

$$PR('A') = (1-d) + d\ (PR('T1')/C('T1') + \ \dots\ + PR('Tn')/C('Tn'))$$

New Improved PageRank Formula:

$$PR('A') = (1-d) + d\ (PR('T1')/[C('T1')-S('T1)] + \ \dots\ + PR('Tn')/[C('Tn')-S('Tn'))$$

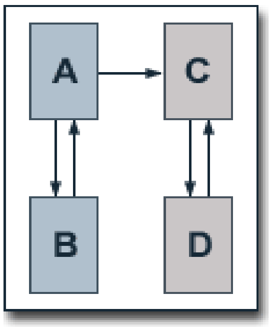
where

- PR('A') is the PageRank of page 'A',
- PR('Ti') is the PageRank of pages 'Ti' which link to page 'A',
- C('Ti') is the number of outbound links on page 'Ti' and
- 'd' is a damping factor which can be set between '0' and '1'.
- S('Ti') is the number of outbound links to same synonyms Title of 'Ti' on page 'Ti'

So, we can see that this PageRank Consider the importance of synonyms outgoing links, Its consider these synonyms outgoing link as similar to link itself, so synonyms related page did not cause to decrease in PageRank value.

Effect of Outbound Links in New Improved PageRank

PageRank is predicated au fait the linking structure of the web; it's ineluctable that if the inward links of a page influence the PageRank, its outward-bound links even have some impact. For example outward-bound links, we are going to take a glance at this straightforward example.



We will take into account an online consisting of 2 websites, every having 2 websites. One website consists of pages ‘A’ and ‘B’; the opposite consists of pages ‘C’ and ‘D’. Initially, each pages of every website link to every different. From this it's obvious that every page incorporates a PageRank of 1. Currently we are going to add a link that wills points from page ‘A’ to page ‘C’. And considering damping issue of ‘0.75’, then we have a tendency to so get following equations for the only pages PageRank values:

PR(‘A’) = 0.25 + 0.75 PR(‘B’)
PR(‘B’) = 0.25 + 0.375 PR(‘A’)
PR(‘C’) = 0.25 + 0.75 PR(‘D’) + 0.375 PR(‘A’)
PR(‘D’) = 0.25 + 0.75 PR(‘C’)

Solving these equations can offers us the subsequent PageRank result values for the primary site:

PR(‘A’) = 14/23
PR(‘B’) = 11/23

We will so get Associate in Nursing accumulated PageRank of ‘25/23’ for this 1st web site. The PageRank values of the second electronic computer square measure given by:

PR(‘C’) = 35/23
PR(‘D’) = 32/23

So, the accumulated add PageRank of second electronic computer is ‘67/23’, the entire PageRank for each sites is ‘92/23’ = four. Hence, adding a link has no

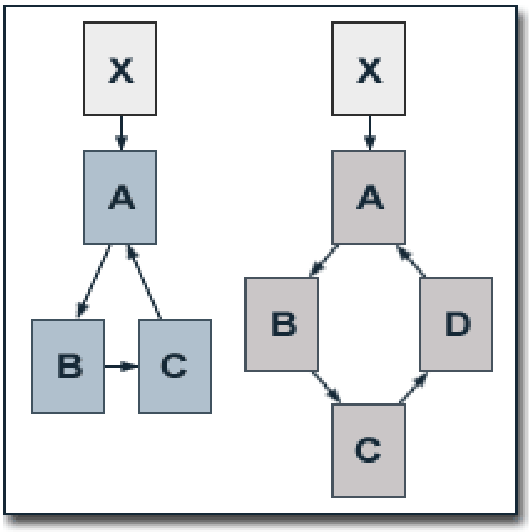
effect on the entire PageRank of the online. Moreover, the PageRank profit for one web site equals the PageRank loss of the opposite web site.
But currently take into account a case that Page ‘A’ and Page ‘C’ are synonyms, therefore currently we are going to take into account this as Page ‘A’ link to itself ‘A’ in situ of ‘C’ therefore during this case No PageRank price will Lose. This explains we have a tendency to be taking care of comparable pages together page. This is the increase the standard of PageRank value.

Intuitive Justification of Effect of Outbound Links

The Logical intuitive justification for the loss of PageRank by a further external departing internet link consistent with the Random internet swimmer Model is that by adding associate external departing net link to 1 page the net swimmer are less seemingly follow an interior link on its web site. So, the likelihood for the net surfer reaching alternative pages inside an internet web site diminishes. If those alternative pages of the internet web site have links back to it page to that the external departing web link has been value-added, conjointly that page's PageRank can exhaust.
So we will conclude that external departing net links diminish the totalized PageRank of an internet web site and possibly conjointly the PageRank of every single website of that site. But, since net links between internet sites are the basics of PageRank and indispensable for its functioning, there's the likelihood that departing net links have positive effects inside the opposite elements of Google's ranking criteria. Lastly, relevant departing links do represent the standard of a page and a webmaster who points to other synonyms web pages means integrates their content in some other way into their own site.

Less Reduction of PageRank by Additional Pages (Improved PageRank algorithm)

By adding websites to a hierarchically structured websites, the implications for the already existing websites are non-uniform. The implications for a websites with a special structure ought to be shown in another example:



We will take a glance at a web site consisting of 3 pages ‘A’, ‘B’ and ‘C’ that are connected to every alternative during a circle. The pages are then joined by page ‘D’ which inserts into this circular linking can structure. This internet site has no outward-bound internet links. Again, links from page ‘X’ that has no alternative outward-bound internet links and a PageRank of ‘10’ points to page ‘A’. At a damping issue (d) of zero.75, the equations for the only pages' PageRank values before adding page ‘D’ square measure given by:

PR(‘A’) = 0.25 + 0.75 (10 + PR(‘C’))
PR(‘B’) = 0.25 + 0.75 * PR(‘A’)
PR(‘C’) = 0.25 + 0.75 * PR(‘B’)

Solving these equations offers us the subsequent PageRank values:

PR(‘A’) = 517/37 = 13.97
PR(‘B’) = 397/37 = 10.73
PR(‘C’) = 307/37 = 8.30

After adding page ‘D’, the equations for the pages' PageRank values are given by:

PR(‘A’) = 0.25 + 0.75 (10 + PR(‘D’))
PR(‘B’) = 0.25 + 0.75 * PR(‘A’)
PR(‘C’) = 0.25 + 0.75 * PR(‘B’)
PR(‘D’) = 0.25 + 0.75 * PR(‘C’)

Solving these equations offers us the PageRank values:

PR('A') = 419/35 = 11.97
PR('B') = 323/35 = 9.23
PR('C') = 251/35 = 7.17
PR('D') = 197/35 = 5.63

Again, when adding page 'D', the accumulated total PageRank of all websites will increase by one from "33 to 34". But now, anybody of the net pages that already existed before page 'D' was side lose the PageRank price. The lot of uniform PageRank is distributed by the net links inside an online web site, the lot of probably this result can occur.
Since adding websites to an online web site usually reduces PageRank for already existing websites, it becomes currently obvious that the PageRank algorithmic rule tends to privilege smaller internet links and sites. Indeed, larger internet sites will counterbalance this result by being a lot of enticing for alternative webmasters (websites) to link to them, just because they need a lot of contents.
None the less, it's additionally doable to extend the PageRank of existing websites by further websites. Therefore, it's to be thought of that as PageRank as doable is distributed to those further pages additionally.

Now consider a case if Most of those Pages are Similar or Synonyms then Reduction is zero or less, its explain that PageRank won't distribute that much to other pages.

Results:

10 Pages Ranking for some Title (From Result)

1)	0.0062129836829642878990632208282	USA
2)	0.00634dd04899942057427778581483732	Microsoft Corporations
3)	0.006328620401002298795452993939772	Microsoft
4)	0.006465017500316119330888022917756	Apple Inc.
5)	0.006637951222194075702608669578516	Delhi
6)	0.006776397060564108018179345430442	America
7)	0.006734816970733471781634500674948	New Delhi
8)	0.006996224297020979024695426845435	Apple
9)	0.007159162479935454599875008279550	Apple Computer
10)	0.007228343207850067270129027345962	Delhi Sultanate

Comparison of PageRank Algorithms

Table of comparisons:

	PageRank(Google)	**Improved PageRank (Ours)**
Description	Divides page rank value equally among all outgoing pages	Divides page rank value equally among outgoing pages, but did not transfer value to Synonyms/Similar page
Based upon	Link structure of Web	Link structure of web with Page Title Keywords
Input	Back and forward links	Back and Forward links, with Synonyms key map
Advantage	Simple, easy	Simple easy
Disadvantage	All outgoing link have reduce value impact	Some outgoing link have reduce value impact

Project Future Plans and Its Extensions

In this Project Report, We show search engine results depend on the various algorithm factors based on this algorithm, web pages are displayed according to their PageRank value which in turn calculated by using many factor like contents, Keywords, Page Title, number of outgoing link etc. Relative comparison of this algorithms is shown above and proposed a hybrid approach of optimizing using Synonyms keywords in Contents And its weighted component need to be subtract and add according to synonyms quality match, genetic algorithms can be useful for search engine to optimization and Synonyms match. This may increase the PageRank value. We can also use hybrid recommender system to suggest synonyms for PageRank calculations.

References:

[1] **Brin, and Larry Page [Google], “The Anatomy of a Large Scale Hyper textual Web Search Engine”. [Reference Paper]**
[2] IEEE
[3] http://hadoop.apache.org/
[4] http://salsahpc.indiana.edu/csci-b649-spring-2014/projects/project3.html
[5] http://www.cs.princeton.edu/~chazelle/courses/BIB/pagerank.htm
[6] http://blog.xebia.com/wiki-pagerank-with-hadoop/
[7] http://pr.efactory.de/