Indian J.Sci.Res. 17(2): 643-648, 2018

ISSN: 0976-2876 (Print)
ISSN: 2250-0138(Online)

PAGE RANKING ALGORITHMS IN INFORMATION RETRIEVA

[1]M.Sowmya,[2]S.Naga Rekha, [3]K.Padmini

[1]Department of Computer Science and Engineering, Aurora Engineering College, Hyderabad, Telangana.

[2]Department of Computer Science and Engineering, Aurora's Technological & Research Institute, .Hyderabad, Telangana.

[3]Department of Computer Science and Engineering, Aurora's Technological & Research Institute, .Hyderabad, Telangana.

Abstract:In order to measure the relative importance of web pages, we propose PageRank, a method for computing a ranking for every web page based on the graph of the web. PageRank has applications in search and browsing estimation.The importance of a Web page is an inherently subjective matter, which depends on the readers interests, knowledge and attitudes. But there is still much that can be said objectively about the relative importance of Web pages. This paper describes Page Rank, a method for rating Web pages objectively and mechanically, effectively measuring the human interest and attention devoted to them. We compare Page Rank to an idealized random Web surfer. We show how to compute Page Rank for large numbers of pages. And, we show how to apply Page Rank to search and to user navigation.Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).Another feature of information retrieval is that it does not actually fetch documents. It only informs the user on the existence and whereabouts of documents relating to his query.

***Keywords*:** InformationRetrieval, Crawling, Selectionpolicy, Re-visitpolicy, Stemmer, Page Rankalgorithm, Invertedindex.

I. Introduction

What is InformationRetrieval

InformationRetrievalistheartofpresentation,storage,organizationofandaccesstoinformation items.Therepresentationandorganizationofinformationshouldbeinsuchawaythattheuser can access information to meet his information need. The definition of information retrieval accordingto(Manningetal.,2009)is:Informationretrieval(IR) isfindingmaterial(usuallydocuments)ofanunstructured nature(usuallytext)thatsatisfiesaninformationneedfromwithinlargecollections (usually stored oncomputers).

Another feature of information retrieval is that it does not actually fetch documents. It only informs the user on the existence and hereabouts of documents relating to his query.

Differencebetweeninformationretrievalanddatare-trieval

The difference between information retrieval and data retrieval is summarized in the following table.

	Data Retrieval	Information Retrieval
Example	Database Query	WWW Search
Matching	Exact	Partial Match, Best Match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Query	Artificial	Natural
Language		
Query Specification	Complete	Incomplete
Items Wanted	Matching	Relevant
Error Response	Sensitive	Insensitive

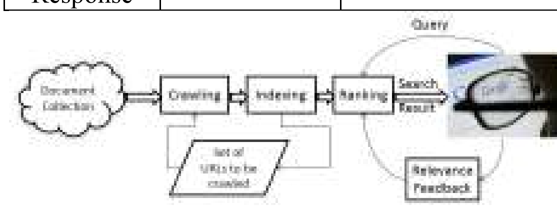

Figure 1.1: Important Processes in Web IR

II. Components of an Information RetrievalSystem

Inthissectionwedescribethecomponentsofabasicwebinformationretrievalsystem.Ageneral information retrieval functions in the following steps. It is shown in Figure1.1.

1. The system browses the document collection and fetches documents. -Crawling
2. Thesystembuildsanindexofthedocuments-Indexing
3. User gives thequery
4. Thesystemretrievesdocumentsthatarerelevanttothequeryfromtheindexanddisplays that to the user -Ranking
5. Usermaygiverelevancefeedbacktothesearchengine-RelevanceFeedback.

[1]Corresponding Author

Thegoalofanyinformationretrievalsystemistosatisfyuser'sinformationneed.Unfortu-nately,characterizationofuserinformationneedisnotsimple.User'softendonotknowclearly abouttheinformationneed.Queryisonlyavagueandincompletedescriptionoftheinformation need. Query operations like query expansion, stop word removal etc. are usually done on the query.

Crawling

Thewebcrawlerautomaticallyretrievesdocumentsfromthewebaspersomedefinedstrategy. The crawler creates a copy of all the documents it crawls to be processed by the search engine. The crawler starts from a list of URLs (documents) called seed. The crawler visits the URLs, identifies the outgoing hyperlinks there and adds them to the list of URLs (documents) to be visited. This way the crawler traverses the web graph following hyperlinks. It saves a copy of each document itvisits.

Selectionpolicy

Selection policy determines which link to crawl first. Generally the web graph is traversed in a breadth first way to avoid being lost at infinite depth. As the number of documents is huge, the selectionstrategybecomescriticalsoastoselectwhichdocumentstocrawlandwhichdocuments nottocrawl.GenerallypageimportancemeasureslikePageRankareusedasaselectionpolicy.

Re-visitpolicy

The crawler needs to crawl frequently to keep the search results up-to-date. The revisit policy determines how frequently the crawling process should be restarted. There is a cost associated withanoutdatedcopyofadocument.Themostlyusedcostfunctionsarefreshness(isthestored copy outdated?) and age (how old is the stored copy).

There may be two revisitpolicies:

Uniform policy Revisit all the documents in the collection with same frequency

Proportional policy Revisit documents that change frequently more often

Itisinterestingtonotethatproportionalpolicyoftenincursmorefreshnesscost.Thereason being,pagesinthewebeitherkeepstaticorchangesofrequentlythateventheproportionalpolicy cannot keep them up todate.

PolitenessPolicy

Beingabot,crawlerscanretrievedocumentsmuchfasterthanhumanusers.Thiswayacrawler caneasilyoverwhelmawebsiteintermsofnetworkresources,serveroverloadetc.anddegrade its performance. The politeness policy restricts a crawler so that these things do not happen. Differentapproachesinthepolitenesspolicyarelistedbelow:

Respecttherobotsexclusionprinciple: Donotcrawltheportionsofthewebpageindicatednottobecrawledintherobots.txtfile.

- DonotcreatemultipleTCPconnectionswiththesameserver
- Introduce a delay between two subsequentrequests

Indexing

The documents crawled by the search engine are stored in an index for efficient retrieval. The documents are first parsed, and then tokenized, stop-word removed and stemmed. After that they are stored in an inverted index. The process is discussed below.

Tokenization

This stem extracts word tokens (index terms) from running text. For example, given a piece of text: "Places to be visited in Delhi" it outputs [places, to, be, visited, in, Delhi].

Stop-wordeliminator

Stop-wordsarethosewordsthatdonothaveanydisambiguationpower.Commonexamplesof stop words are articles, prepositions *etc.* In this step, stop words are removed from the list of tokens. For example, given the list of token generated by tokenizer, it strips it down to: [places, visited,Delhi].

Stemmer

Theremainingtokensarethenstemmedtotherootform(*e.g*visit_visited). For example, afterstemmingthelistoftokensbecomesthis:[place,visit,Delhi].

Invertedindex

Theordinaryindexwouldcontainforeachdocument,theindextermswithinit.Buttheinverted index stores for each term the list of documents where they appear. The benefit of using an inverted index comes from the fact that in IR we are interested in finding the documents that contain the index terms in the query. So, if we have an inverted index, we do not have to scan through all the documents in collection in search of the term. Often a hash-table is associated withtheinvertedindexsothatsearchinghappensinO(1)time.

Inverted index may contain additional information like how many times the term appears in the document, the offset of the term within the document etc.

Example Say there are three documents.

Doc1 Milk is nutritious

Doc2 Bread and milk tastes good

Doc1 Brown bread is better

After stop-word elimination and stemming, the inverted index looks like:

milk	1,2
nutritious	1
bread	2,3
taste	2
good	2
brown	3
better	3

Ranking

When the user gives a query, the index is consulted to get the documents most relevant to the query.Therelevantdocumentsarethenrankedaccordingtotheir degreeofrelevance,importance etc. Ranking is discussed elaborately in the subsequentchapters.

RelevanceFeedback

Relevancefeedbackisoneoftheclassicalwaysofrefiningsearc henginerankings.Itworksinthe following way: Search engine firsts generate an initial set of rankings. Users select the relevant documentswithinthisranking.Basedontheinformationinthes edocumentsamoreappropriate rankingispresented(forexample,thequerymaybeexpandedus ingthetermscontainedinthe first set of relevantdocuments).

Sometimesusersdonotenoughdomainknowledgetoformgoo dqueries.Buttheycanselectrelevantdocumentsfromalistofdo cumentsoncethedocumentsareshowntohim. Forexample, when the user fires a query 'matrix', initially documents on both the topics (movie and maths) are retrieved. Then say, the user selects the maths documents as relevant. This feedback can be used to refine the search and retrieve more documents from mathematicsdomain.

Types of relevancefeedback

Explicit User gives feedback to help system to improve.

Implicit User doesn't know he is helping *e.g*"similar pages" features in Google.

Pseudo User doesn't do anything! Top 'k' judgments are taken as relevant. Being fully au- tomated it has always this risk that results may drift completely away from the intended document set.

Issues with relevancefeedback

The user must have sufficient knowledge to form the initialquery.

Thisdoesnotworktoowellincaseslike:Misspellings,CLIR,an dMismatchinuser'sand document's vocabulary (Burma vs.Myanmar).

Relevant documents has to be similar to each other (they need to cluster) while similarity betweenrelevantandnon-relevantdocumentshouldbesmall.Thatiswhythistechnique does not work too well for inherently disjunctive queries (Pop stars who once worked at Burger King) or generic topics (tigers) who often appear as disjunction of more specific concepts.

Longqueriesgeneratedmaycauselongresponsetime.

Users are often reluctant to participate in explicit feedback. [Spink et al. (2000): Only 4% users participate. 70% doesn't go beyond first page.]

Inweb,clickstreamdatacouldbeusedasindirectrelevancefeed back(discussedinautore- fchapter: conclusion).

III. Searching the Web: LinkAnalysis and Anchor Text

In the last chapter we discussed theoretical models of IR and their practical applications. They all share a commonality. All the models rank the documents based on the similarity of them with the query and for doing this they use features from the document only. This approach wassuitableforinformationretrievalfromawell-controlledcollectionofdocumentslikeresearch papers or library catalog. But the scenario in the case of Web Search is substantiallydifferent.

Difference between Web IR and TraditionalIR

But in the case of ranking of web documents, using features only within the document is not sufficient. Onereasonis,asthenumberofdocumentsisveryhugeincaseof web,alargenumber documents are often very relevant to the query which cannot be further ranked based only on the internal features of thedocument.

As the web is growing really fast, the search process must be scalable, algorithms must be efficientandstorageshouldbeusedproperly,queriesshouldbe handledquickly(hundredsof thousands per second). In web, thousands of documents would match the query but only the top few are those who count. A Web IR system must avoid junk results at top (Brin and Page,1998).So,"veryhighprecisionisimportantevenattheexp enseofrecall".

The web is a huge set of totally uncontrolled heterogeneous set of documents. There aremanylanguagesandmanydifferentdocumentformatsinvol ved.Anyonecanpublishanything in the web. Not all documents are of same importance. For example, a random blog and the BBCwebsitecannotbetreatedinthesameway.

Searchenginestakeaverybigroleinroutingtraffictowardsawe bpage.Asthereisvirtually no control over what people can put on the web, a malicious user may put random interesting wordsinhispage(probablydothingstomakethesetermsnoteasilyvisibletothevisitor)and get a high ranking in any term-frequency based ranking method. Meta fields like "keywords" are often used for this purpose as they are not directly visible. This makes clear that we cannot relyonlyonthedocumenttogetitsrank.

In this chapter we discuss two features specific to web that can be exploited to carry on Web IR. One is Link Analysis and another is Anchor Text.

LinkAnalysis

Fortunately, web gives us a unique way tomeasure the importance of a document. In web, documents are often connected using hyperlinks. If a page B has a link to page A, then we say pageAhasabacklinkfrompageB. We canviewback-linksasatypeofendorsement.Themore backlinksapagehave,themoreimportantthepageis.Whilerankingiftwopageshavesimilar relevance to the query, the more important page should be ranked higher. In the next section we formalize the notions.

Web as agraph

Web is a collection of hyperlinked documents. We can view the web as a directed graph where each page[1]is a node and a hyperlink from one page to another is captured by a directed edge

. If page A has a link to page B, then there should be a directed edge from node A to node B. Every page has a number of forward edges (out edges) and backlinks (in edges). Page A has a backlink from page B if there is a hyperlink to page A from page B. Backlink can be viewed as a type of endorsement and the count of backlinks is regarded as a measure of importance of a page. This idea was used earlier in the field of citation analysis.

But deciding the importance of a page based on backlink count pose another problem in terms of link farms. Link farms are a group of web pages that link to each other. In this way anymaliciouscreatorofwebpagecanhavehighbacklinkcountbysettingupanothernnumber ofwebpageseachhavingahyperlinktothatpage.Thealgorithm PageRanksolvesthisproblem bynotonlycountingthebacklinksbutalsonotingthepagefrom whichthelinkiscomingfrom.

PageRankalgorithm

PageRankisalinkanalysisisalgorithmthatestimatestheimportanceofadocumentbyanalyzing the link structure of a hyperlinked set of documents. It is named after Larry Page (co-founder ofGoogle).

Thesimplebacklinkcountwassufficientforwellcontrolleddocumentcollectionofcitation analysis.Butinweb,thereishardlyanycontrol.Millionsofpagescanbeautomaticallycreated andlinkedwitheachothertomanipulatethebacklinkcount(Pageetal.,1998).Aswebconsists ofconflictingprofitmakingventures,anyevaluationstrategywhichcountsreplicablefeaturesofwebpagesisboundtobemanipulated.

PageRank extends the idea of backlink by "not counting links from all pages equally, and bynormalizingbythenumberoflinksonapage."(BrinandPage, 1998).Here,weassumepage$A$ has pages $T_1 \ldots T_n$which point to it (*i.e.*, are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. Also $C(A)$ is defined as the number of links going out of page A (a normalizing factor). The PageRank of a page A is a value in the range0to1andisgivenby$PR(A)=\frac{1-d}{N}+d\,\Sigma\,PR(T_1)+\cdots+PR(T_n)$

Calculation ofPageRank

Although the expression for PageRank is a recursive one, it is calculated in an iterative way. At first all the pages are given uniform PageRank (= $1/N$ where N is the number of pages in collection). With every iteration, the PageRank values are approximated by Equation 3.1.After apointoftimetheprocessconverges.

Disadvantage ofPageRank

PageRankfavorsolderpagesthannewerones.Theolderpagesareexpectedtohavemorenumber of citations from important page than a page just introduced. Therefore, page rank should not beusedasastandalonemetric.Itshouldbeusedasaparameteronly.

Different Types of Page Ranking Algorithms:

1.The First-In, First-Out (FIFO) Page Replacement Algorithm

Another low-overhead paging algorithm is the **FIFO** (**First-In, First-Out**) algorithm. To illustrate how this works, consider a supermarket that has enough shelves to display exactly *k* different products. One day, some company introduces a new convenience food—instant, freeze-dried, organic yogurt that can be reconstituted in a microwave oven. It is an immediate success, so our finite supermarket has to get rid of one old product in order to stock it.

One possibility is to find the product that the supermarket has been stocking the longest (i.e., something it began selling 120 years ago) and get rid of it on the grounds that no one is interested any more. In effect, the supermarket maintains a linked list of all the products it currently sells in the order they were introduced. The new one goes on the back of the list; the one at the front of the list is dropped.

As a page replacement algorithm, the same idea is applicable. The operating system maintains a list of all pages currently in memory, with the page at the head of the list the oldest one and the page at the tail the most recent arrival. On a page fault, the page at the head is removed and the new page added to the tail of the list. When applied to stores, FIFO might remove mustache wax, but it might also remove flour, salt, or butter. When applied to computers the same problem arises. For this reason, FIFO in its pure form is rarely used.

2.The Second Chance Page Replacement Algorithm

A simple modification to FIFO that avoids the problem of throwing out a heavily used page is to inspect the *R* bit of the oldest page. If it is 0, the page is both old and unused, so it is replaced immediately. If the *R* bit is 1, the bit is cleared, the page is put onto the end of the list of pages, and its load time is updated as though it had just arrived in memory. Then the search continues.

The operation of this algorithm, called **second chance**, is shown in Fig. -1(a) we see pages *A* through *H* kept on a linked list and sorted by the time they arrived in memory.

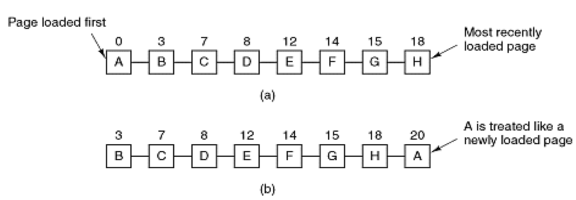

Figure 2.Operation of second chance. (a) Pages sorted in FIFO order. (b) Page list if a page fault occurs at time 20 and A has its R bit set. The numbers above the pages are their loading times.

Suppose that a page fault occurs at time 20. The oldest page is *A*, which arrived at time 0, when the process started. If *A* has the *R* bit cleared, it is evicted from memory, either by being written to the disk (if it is dirty), or just abandoned (if it is clean). On the other hand, if the *R* bit is set, *A* is put onto the end of the list and its "load time" is reset to the current time (20). The *R* bit is also cleared. The search for a suitable page continues with *B*.

What second chance is doing is looking for an old page that has not been referenced in the previous clock interval. If all the pages have been referenced, second chance degenerates into pure FIFO. Specifically, imagine that all the pages in Fig. 1-1(a) have their *R* bits set. One by one, the operating system moves the pages to the end of the list, clearing the *R* bit each time it appends a page to the end of the list. Eventually, it comes back to page *A*, which now has its *R* bit cleared. At this point *A* is evicted. Thus the algorithm always terminates.

3.The Clock Page Replacement Algorithm

Although second chance is a reasonable algorithm, it is unnecessarily inefficient because it is constantly moving pages around on its list. A better approach is to keep all the page frames on a circular list in the form of a clock, as shown in **Fig 2.** A hand points to the oldest page.

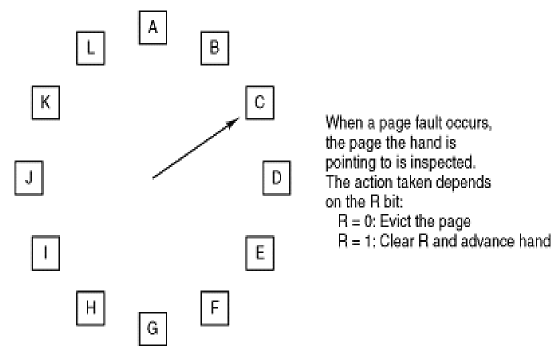

Figure 3.The clock page replacement algorithm.

When a page fault occurs, the page being pointed to by the hand is inspected. If its *R* bit is 0, the page is evicted, the new page is inserted into the clock in its place, and the hand is advanced one position. If*R* is 1, it is cleared and the hand is advanced to the next page. This process is repeated until a page is found withR = 0. Not surprisingly, this algorithm is called **clock**. It differs from second chance only in the implementation.

References

[1]. Sergey Brin and Larry Page. oogle search engine. http://google.stanford.edu.[CGMP98]

[2]. JunghooCho,HectorGarcia-Molina,andLawrencePage.Ecientcrawlingthroughurlordering. In To Appear: Proceedings of the Seventh International Web Conference1998

[3] .Stephen Robertson (Microsoft Research Cambridge) and Hugo Zaragoza (Yahoo! Research Barcelona).Theprobabilisticrelevancemethod: Bm25andbeyond-slidesfromsigir2007. 2007.http://www.zaragozas.info/hugo/academic/pdf/tutorial_sigir07_2d.pdf.

[4]. Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):3543, 2001.

[5]. Brin and L. Page. The anatomy of a large-scale

hypertextual web search engine.*Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[6] L.Page,S.Brin,R.Motwani,andT.Winograd,"ThePagerankCitationRanking:Bringingorder to the Web". Technical Report, Stanford Digital Libraries SIDL-WP1999-0120,1999.google search engine. http://google.stanford.edu.[CGMP98]

[7] JunghooCho,HectorGarcia-Molina,andLawrencePage.Ecientcrawlingthrough url ordering. In To Appear: Proceedings of the Seventh International Web Conference1998

[8 .Stephen Robertson (Microsoft Research Cambridge) and Hugo Zaragoza (Yahoo! Research Barcelona).Theprobabilisticrelevancemethod: Bm25andbeyond-slidesfromsigir007.2007.http://www.zaragozas.info/hugo/academic/pdf/tutorial_sigir07_2d.pdf