computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

SBLWPR – SIMILARITY BASED LINK WEIGHT FOR PAGERANK CALCULATION

S. Poomagal, T. Hamsapriya

PSG College of Technology,
Coimbatore, India
e-mail: poomagal_sam@yahoo.co.in

Abstract: *Search engine retrieves list of web pages which are relevant to the given query from the index and sorts the list based on the page importance score. There are different ranking algorithms available in the literature to calculate the importance score of web pages. The basis of all ranking algorithms is the link structure of the web. In existing ranking algorithms, no weight is assigned to the links by considering the similarity among the linked documents. Since links from similar documents are more important than the links from other dissimilar documents, a new method is introduced to assign weight to each link based on the similarity among the linked documents. Calculated link weight is added with existing PageRank value to calculate final PageRank. Proposed technique is compared with existing ranking algorithms using the measures precision, recall and F-measure.*

Keywords: *Authority Score, Hub Score, Link structure, PageRank, Similarity, Stemming.*

1. INTRODUCTION

Web mining is used to search the content of the Web, to perform link analysis and to identify the users' behavior in the past to predict the future usage of the web. Based on the above, web mining is divided into three categories such as Web Content Mining (WCM), Web Structure Mining (WSM), and Web Usage Mining (WUM) [1, 2, 3].

WCM discovers useful information from the web document content by applying some traditional data mining techniques. WSM deals with the discovery of relationships between web pages by analyzing web hyperlink structure. WUM mines user log files to identify the users' behavior in viewing the web pages. This information is helpful to make future decisions.

All existing search engines perform Web Structure Mining using inlinks and outlinks of the web pages to identify the popularity of a page. Based on the popularity, ranks are assigned to the web documents. A page is more popular if it is pointed by many pages. Using this concept as a base, many algorithms were devised to rank the pages according to its importance.

One such technique is PageRank [4] used by Google search engine and it has proved to be a very effective algorithm for finding the rank of the search results.

Improvement in the PageRank algorithm is done by HITS (Hypertext Induced Topic Selection)[5] using the concept of authorities and hubs. Authoritative pages have more number of incoming links and hub pages have more number of outgoing links. [6, 7, 8, 9] first identifies pages of interest through term-based techniques and then performs an analysis of only the graph neighborhood of these pages. Major problems with HITS algorithm is link spamming and topic drift [10].

Another method called SALSA (Stochastic Approach for Link Structure Analysis) introduced by [11] combines the random walk method of PageRank with hub and authority technique of HITS. It eliminates the drawbacks of HITS such as link spamming and topic drift [10].

Wenpu Xing and Ali Ghorbani have introduced Weighted PageRank algorithm [12] which assigns larger rank values to more popular pages instead of dividing the rank value of a page evenly among its outlink pages. Each outlink page gets a value proportional to its popularity.

In all the above ranking algorithms, links are not assigned with a weight value according to the similarity among the documents that are linked. In this paper, each link is assigned with a weight based on the level of similarity among the linked documents. Calculated link weight is added with the existing PageRank value.

First step of the proposed method is to extract terms from the documents. After extracting the terms, stop words are removed. Stop words are the terms that do not have any meaningful information to find the similarity among the documents. Resultant terms after stop word removal will undergo the process called stemming. Stemming is the process of converting the terms into their base forms. Stemmed terms are collected and TFIDF (Term Frequency / Inverse Document Frequency) value of the terms in every document is calculated and filled in the vector space. Vector space includes set of documents (D1 to DM) as rows and the collection of terms (T1 to TN) as columns. Distance matrix is formed from the TFIDF matrix using Euclidean distance function. Using the distance matrix, weight of each link is calculated and added with the initial PageRank.

The remainder of this paper is organized as follows: Section 2 discusses the ranking algorithms available in the literature. Section 3 explains the proposed method. Section 4 discusses the results obtained. Section 5 concludes the paper and Section 6 mentions the future work.

2. EXISTING RANKING ALGORITHMS

One of the major challenges in information retrieval is the ranking of search results. In the context of web search, where the data is massive and queries rarely contain more than three terms, most searches produce large collection of results. Since the majority of search engine users examine only first few pages of search results [13], effective ranking algorithms are necessary for satisfying users' needs by bringing more relevant documents to first few pages. Many research works were done in link based ranking algorithms. Most of the research works has centered on proposing new link based ranking algorithms or improving the efficiency of existing ones.

The following are the popular ranking algorithms in the literature:

i. PageRank
ii. HITS
iii. SALSA
iv. Weighted PageRank

2.1. PAGERANK

PageRank algorithm [4] is a commonly used algorithm which does Web Structure Mining. PageRank is used to sort the results so that more relevant pages are likely to be displayed at the beginning of the list of search results. It measures the importance of the pages by analyzing the links through markov chain model [14].

The fundamental principle of PageRank is that "a page is important, if it being pointed by many other pages". Hence, to quantify the PageRank value of a page A, the sum of the PageRank of other pages pointing to page A is computed. It is described mathematically as,

$$r(P_i) = \sum_{p_j} \frac{r(P_i)}{|p_j|} \tag{1}$$

where P_j denotes the set of all pages in the web hyperlink structure that points to the page $P_{i.}$ $|P_j|$ is the number of outlinks of P_i. This equation computes the PageRank of the pages one at a time. However, a careful analysis will reveal that matrix multiplication can be used to compute the entire PageRank vector at a time. To avoid a page dominating the PageRank values of other pages and to suppress bogus pages, the rows of the matrix are normalized by dividing the entries with the total number of outlinks of the page. Mathematically, the iteration is defined as

$$r(P_i) = \sum_{p_j} \frac{r(P_i)}{|p_j|} P^{T(k+1)} = P^{T(k)} H \tag{2}$$

where k denotes the number of iterations, P is the PageRank vector and H is the matrix obtained after normalizing the rows. After normalizing, the matrix becomes substochastic and we know the matrix thus obtained is the power method for left hand eigenvector computation and we also know that a stochastic matrix will have a stationary eigenvector. Hence the normalized hyperlink matrix is converted to a stochastic matrix by replacing the row entries, of the pages with no outlinks, with 1/n, where n is the number of pages. Mathematically, it is given as,

$$P^{T(k+1)} S = H + a\left(\frac{1}{n} c^T\right) \tag{3}$$

where 'c' is a column vector of all 1's and is of order nx1, 'a' is a column vector which has '1' entry for the dangling nodes (pages with no outlinks) and S is the stochastic matrix. However, the problem that still persists is the rank sink problem which is characterized by a single page or a set of pages dominating the PageRank values of other pages. Some pages may obtain a rank of zero in the process of convergence which is not conceptually possible. To overcome this problem, a teleportation matrix is used in the computation of the rank vector which also preserves the nature of the original input. The computations in creating the teleportation matrix can be mathematically modeled as

$$G = \alpha S + (1-\alpha)\frac{1}{n}cc^T \tag{4}$$

For an optimum mixture of the original values, α value is taken as 0.85. The iterations with the G matrix will provide a converged, unique PageRank vector for all the pages irrespective of the initial vector. Based on the PageRank vector, the pages are ranked and sorted.

2.2. HITS

HITS (Hypertext Induced Topic Selection) [5] algorithm calculates two different scores namely hub score and authority score. It collects pages and forms a graph with authorities and hubs. Web pages pointed by many hyperlinks are called authorities. Web pages that point to many other pages are called hubs. Strong authority is the page which has links from many highly scored hubs. Popular hub is the page which points or links to highly scored authorities.

Authority score of a page is a function of the sum of the hub scores of the pages pointing to it while the hub score of a page is given by the sum of authority scores of the pages that points to it. These calculations can be performed through matrix multiplications using the adjacency matrix.

Steps

1. **Construct an adjacency matrix by using the neighborhood graph N which indicates the connectivity of all the nodes.**

2. **Calculate the authority vector from the adjacency matrix formed using the formula**

$$V_k = \left(A^T A\right) \tag{5}$$

3. **Calculate the hub vector from the adjacency matrix using the formula**

$$U_k = \left(AA^T\right)V_k \tag{6}$$

4. **Rank the pages using the hub and authority vectors formed.**

The equations 5 & 6 also define the iterative power method for computing the dominant eigenvector as in the PageRank algorithm. However, the implementation of HITS differs from the method used in the PageRank algorithm. HITS algorithm is query dependent and it processes only the pages that correspond to the given query either directly or indirectly.

Initially, the base set is constructed by using the pages that directly correspond to the query and then it is expanded by adding the inlinks and outlinks of the pages in the base set. The set of pages thus obtained is visualized in the form of a web hyperlink graph and the adjacency matrix is constructed. The authority score and the hub score of a page is calculated as given in equations 5 and 6. The score vector is normalized by dividing the score of each page with the maximum score in that iteration. The matrices used are symmetric, positive semi definite, nonnegative and hence ensure convergence.

Major advantage of HITS algorithm is its dual rankings. HITS presents two ranked lists to the user. One with more authoritative documents and the other one with most hubby documents. Authority score can be used when the search is oriented towards research. Hub score can be used when the search is broad. Another advantage of HITS is the size of the problem. It casts the ranking problem as a small problem, finding the dominant eigenvectors of small matrices. The size of these matrices is very small relative to the total number of pages on the web.

Major disadvantage of HITS algorithm is susceptibility to link spamming [10]. By adding links to and from any web page, it is possible to change the hub and authority scores. Since hub score and authority score are interdependent, when hub score is increased by introducing more outlinks on a page, automatically the authority score of a page increases. Another problem with HITS is topic drift [10]. In building neighborhood graph N for a query it is possible that a very authoritative yet off-topic page be linked to a page containing the query terms. This very authoritative page can carry so much weight and its neighboring documents dominate the relevant ranked list returned to the user, skewing the results toward off-topic documents. Another drawback of original HITS algorithm is that it is query dependent. At query time, a neighborhood graph must be constructed and at least one eigenvector problem must be solved. This problem can be rectified by making HITS query-independent. It can be done by dropping the neighborhood graph step and computing the authority and hub vectors using the adjacency matrix of the entire web graph.

2.3. SALSA

SALSA, the Stochastic Approach for Link-Structure Analysis [11] is based on the theory of Markov chains, and uses the stochastic properties of random walks done on a collection of pages along with hub and authority technique of HITS. The meta algorithm used by both HITS and SALSA is similar but the basic difference between two methods is the formation of an adjacency matrix. HITS algorithm considers the tight connection between the nodes of the graph but SALSA considers light connection by performing random walk in the graph.

Initially the neighborhood graph N associated with a particular query is formed. SALSA differs from HITS in the next step. Rather than forming an adjacency matrix L for the neighborhood graph N, a bipartite undirected graph G is built. G is defined by three sets: V_h, V_a and E, where V_h is a set of hub nodes and V_a is a set of authority nodes and E is a set of directed edges in N. Next, two Markov chains are formed from G, a hub Markov chain with transition probability matrix H, and an authority Markov chain with matrix A.

HITS used the adjacency matrix L of N to compute authority and hub scores. On the other hand, PageRank computes a measure analogous to an authority score using a row-normalized weighted matrix G. SALSA uses both row and column weighting to compute its hub and authority scores. Let L_r be L with each nonzero row divided by its row sum and L_c be L with each nonzero column divided by its column sum.

Steps

1. **A bipartite graph is drawn with hubs in one side of the graph and authorities in another of the graph.**
   i. Hub includes the nodes (V_h) with outdegree greater than zero
   and
   ii. Authority side includes nodes (V_a) with indegree greater than zero
2. **Column and row weighted matrices L_c and L_r are formed.**
3. **Hub and authority matrices are formed by**

$$A = L_r L_c^T \qquad \textbf{(7)}$$

$$A = L_c^T L_r \qquad \textbf{(8)}$$

where L_r – Non-zero rows of L divided by its row sum
L_c – Non-zero rows of L divided by its column sum

4. **Eigenvectors are formed from the hub and authority matrices.**
5. **Based on the hub and authority vectors, the pages are ranked.**

SALSA is less susceptible to link spamming [10] since the interdependence between hub and authority scores is much less. Unlike HITS, SALSA is victimized by the topic drift [10] problem. Serious drawback of SALSA is its query dependence. At query time, the neighborhood graph N for the query must be formed and the stationary vectors for two Markov chains must be computed. Another problem with SALSA is convergence. Since SALSA does not force irreducibility onto the graph, the resulting vectors produced by the algorithm may not be unique if the neighborhood graph is reducible.

2.4. WEIGHTED PAGERANK

It assigns larger rank values to more important (popular) pages instead of dividing the rank value of a page evenly among its outlink pages. Each outlink page gets a value proportional to its popularity (its number of inlinks and outlinks). The popularity from the number of inlinks and outlinks is recorded as $W^{in}_{(v,u)}$ and $W^{out}_{(v,u)}$, respectively.

$W^{in}_{(v,u)}$ is the weight of link (v, u) calculated based on the number of inlinks of page u and the number of inlinks of all reference pages of page v.

$$W^{in}_{(v,u)} = \frac{I_u}{\sum_{p\in R(v)} I_p} \qquad (9)$$

where I_u and I_p represent the number of inlinks of page u and page p, respectively. R(v) denotes the reference page list of page v.

$W^{out}_{(v,u)}$ is the weight of link(v, u) calculated based on the number of outlinks of page u and the number of outlinks of all reference pages of page v.

$$W^{out}_{(v,u)} = \frac{o_u}{\sum_{p\in R(v)} o_p} \qquad (10)$$

where O_u and O_p represent the number of outlinks of page u and page p, respectively. R(v) denotes the reference page list of page v.

PageRank formula is modified as

$$PR(u) = (1-d) + \sum_{v\in B(u)} PR(v)\, W^{in}_{(v,u)} W^{out}_{(v,u)} \qquad (11)$$

3. PROPOSED METHOD

Proposed method assigns ranks to the pages by calculating the weight of each link based on the similarity among the documents connected by that link. Weight is added with the existing PageRank formula to produce final PageRank.

Steps in the Proposed Method

1. Term Extraction

2. Pre-Processing (Stop word removal and Stemming)

3. TFIDF Matrix formation

4. Distance Calculation

5. Weight Calculation

6. Final PageRank Calculation

3.1. TERM EXTRACTION

Text documents are different from web documents. Web documents are unstructured. In addition to text contents it also contains tag information. This tag information and the punctuations should be removed from the document to extract meaningful terms. Tokenization is used to perform this operation.

Given a document, tokenization is the task of breaking it into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation. Here is an example of tokenization:

Input: Friends, Romans, Country men, lend me your ears;

Output: Friends Romans Country men lend me your ears

Contents from the documents are extracted by removing the tags and special symbols with the use of tokenization. Extracted terms are given as an input to the next step.

3.2. PREPROCESSING (STOP WORD REMOVAL AND STEMMING)

Stop words does not provide any useful information to identify the similarity among the pages. So they can be removed to avoid confusions. Some common stop words are is, was, are, were, what etc., For stop word removal, initially the database of stop words is created and the terms extracted from the web pages are compared with the database of stop words. Stop words found in the extracted collection of terms are removed.

The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

am, are, is $\Rightarrow$ be
car, cars, car's, cars' $\Rightarrow$ car
the boy's cars are different colors $\Rightarrow$ the boy car be differ color

Stemming usually refers to a crude heuristic process that chops off the ends of words and often includes the removal of derivational affixes.

Usually the stemmed words are not meaningful. For example, the stemmed word of “computation” is “comput”. While stemming the words, two things have to be considered.

i. Different words with the same base meaning are converted to the same form

and

ii. Words with distinct meanings are kept separate.

For stemming, Porter’s algorithm is used in this paper. It is a simple utility that reduces English words to their word stems – without the “ing”, “ings”; “s” etc.,

Following table shows the sample stemmed words:

Table 1. Words and their equivalent base words

Word	Base Word	Word	Base Word
Consigned	Consign	Consolation	Consol
Consigning	Consign	Consolations	Consol
Consignment	Consign	Console	Consol
Consisted	Consist	Consoled	Consol
Consistency	Consist	Consoles	Consol
Consistently	Consist	Consonant	Conson
Consisting	Consist	Consorted	Consort
Consists	Consist	Conspirator	Conspir

3.3. TFIDF CALCULATION

TFIDF is frequently used to construct a term vector space model. It evaluates the importance of a word in a document. The importance score increases proportionally with the number of times a word appears in the document but is offset by the frequency of a word in the entire collection of documents. Suppose there are set of documents, each with collection of terms. A simple way to find the similarity among those documents using terms is to count the number of times a term occurs in a document. Calculated count is called as a term frequency. However, some terms are more common such as “contain” and these terms get more weight when term frequency is used. Also the terms like “contain” are not good keywords to identify the similarity among the documents. On the other side, the keywords that occur rarely are good to find the relevancy among the documents. Hence an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the collection and increases the weight of the terms that occur rarely.

This assigns to term i a weight in document j given by

$$TFIDF_{i,j} = TF_{i,j} X IDF_i \qquad (12)$$

$TF_{i,j}$ is calculated as:

$$TF_{i,j} = \frac{N_{i,j}}{NT_j} \tag{13}$$

$N_{i,j}$ is the number of times the term i appears in the document j and NT_j is the total number of terms in the document j.

The inverse document frequency (IDF_i) is calculated as:

$$IDF_i = \log\left(\frac{|D|}{|dt_i \in d|}\right) \tag{14}$$

where |D| is the total number of documents and $|d : t_i \in d|$ is the number of documents in which the term t_i appears. These TFIDF values and the list of documents are then formed as a vector space.

Term / Document matrix is as follows,

Docs / Terms	T_1	T_2	…….	T_N
D_1	$TFIDF_{11}$	$TFIDF_{12}$	…….	$TFIDF_{1N}$
D_2	$TFIDF_{21}$	$TFIDF_{22}$	…….	$TFIDF_{2N}$
.				
D_M	$TFIDF_{M1}$	$TFIDF_{M2}$	….	$TFIDF_{MN}$

where N denotes the number of terms and M denotes the number of documents.

3.4. DISTANCE CALCULATION

From the TFIDF matrix, distance from every document to every other document in the collection is calculated using the Euclidean distance function and distance matrix is formed. Euclidean distance formula is,

$$Dist_{i,j} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{15}$$

where i changes from 1 to N. N is the number of terms in the vector space.

Following shows the Distance matrix.

Documents	D_1	D_2	…….	D_M
D_1	$DIST_{11}$	$DIST_{12}$	…….	$DIST_{1M}$
D_2	$DIST_{21}$	$DIST_{22}$	……..	$DIST_{2M}$
.				
D_M	$DIST_{M1}$	$DIST_{M2}$	……...	$DIST_{MM}$

3.5. WEIGHT CALCULATION

Weight for each link is calculated using the distances from the distance matrix. Weight is calculated using the following formula,

$$W_{ij} = \frac{M}{Dist_{ij}} \tag{16}$$

where M is the number of documents, $Dist_{ij}$ is the distance between the documents i and j and the values of i and j varies from 1 to M.

After calculating the weight of each link, weight matrix is formed as follows,

Documents	D_1	D_2	…….	D_M
D_1	W_{11}	W_{12}	…….	W_{1M}
D_2	W_{21}	W_{22}	…….	W_{2M}
.				
D_M	W_{M1}	W_{M2}	…….	W_{MM}

3.6. PAGERANK CALCULATION

Existing PageRank formula is modified to include the calculated weight as,

$$PR(D_i) = (1-d) + d\sum\nolimits_{j\in L} PR(D_j) + W_{ij} \tag{17}$$

where D_i represents the document for which the PageRank is to be calculated, D_j represents the document which has out-link to D_i and L represents the numbers of the documents which has out-link to D_i. W_{ij} represents the weight of a link between the documents i and j.

4. EXPERIMENTAL RESULTS

For our experiment, 200 queries are considered. For all 200 queries, first 200 results from yahoo, google and bing are retrieved. Code is written using JAVA and SQL Server is used to store the collected web documents. The measures such as precision, recall and F-measure are used to compare the proposed method with the existing methods.

Initially, the words are extracted from the web documents and stop words are removed. Table 2 shows the stop words considered in this paper.

Table 2. List of Stop words

a	a, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, ain't, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways,

	anywhere, apart, appear, appreciate, appropriate, are, aren’t, around, as, aside, ask, asking, associated, at, available, away, awfully
b	be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, believe, below, beside, besides, best, better, between, beyond, both, brief, but, by
c	common, came, can, can’t, cannot, cant, cause, causes, certain, certainly, changes, clearly, co, com, come, comes, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, couldn’t, course, currently
d	definitely, described, despite, did, didn’t, different, do, does, doesn’t, doing, don’t, done, down, downwards, during,
e	each, edu, eg, eight, either, else, elsewhere, enough, entirely, especially, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, exactly, example, except
f	far, few, fifth, first, five, followed, following, follows, for, former, formerly, forth, four, from, further, furthermore
g	get, gets, getting, given, gives, go, goes, going, gone, got, gotten, greetings
h	had, hadn’t, happens, hardly, has, hasn’t, have, haven’t, having, he, he’s, hello, help, hence, her, here, here’s, hereafter, hereby, herein, hereupon, hers, herself, hi, him, himself, his, hither, hopefully, how, howbeit, however
i	i’d, i’ll, i’m, i’ve, ie, if, ignored, immediate, in, inasmuch, inc, indeed, indicate, indicated, indicates, inner, insofar, instead, into, inward, is, isn’t, it, it’d, it’ll, it’s, its, itself
j	Just
k	keep, keeps, kept, know, knows, known
l	last, lately, later, latter, latterly, least, less, lest, let, let’s, like, liked, likely, little, look, looking, looks, ltd
m	mainly, many, may, maybe, me, mean, meanwhile, merely, might, more, moreover, most, mostly, much, must, my, myself
n	name, namely, nd, near, nearly, necessary, need, needs, neither, never, nevertheless, new, next, nine, no, nobody, none, nor, normally, not, nothing, novel, now, nowhere
o	obviously, of, off, often, oh, ok, okay, old, on, once, one, ones, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, own
p	particular, particularly, per, perhaps, placed, please, plus, possible, presumably, probably, provides
q	que, quite, qv
r	rather, rd, re, really, reasonably, regarding, regardless, regards, relatively, respectively, right
s	said, same, saw, say, saying, says, second, secondly, see, seeing, seem, seemed, seeming, seems, seen, self, selves, sensible, sent, serious, seriously, seven, several, shall, she, should, shouldn’t, since, six, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, soon, sorry, specified, specify, specifying, still, sub, such, sup, sure
t	t’s, take, taken, tell, tends, th, than, thank, thanks, thanx, that, that’s, thats, the, their, theirs, them, themselves, then, thence, there, there’s, thereafter, thereby, therefore, therein, theres, thereupon, these, they, they’d, they’ll, they’re, they’ve, think, third, this, thorough, thoroughly, those, though, three, through, throughout, thru, thus, to, together, too, took, toward, towards, tried, tries, truly, try, trying, twice, two
u	un, under, unfortunately, unless, unlikely, until, unto, up, upon, us, use, used, useful, uses, using, usually
v	value, various, very, via, viz, vs
w	want, wants, was, wasn’t, way, we, we’d, we’ll, we’re, we’ve, welcome, well, went, were, weren’t, what, what’s, whatever, when, whence, whenever, where, where’s, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, who’s, whoever, whole, whom, whose, why, will, willing, wish, with, within, without, won’t, wonder, would, would, wouldn’t
y	yes, yet, you, you’d, you’ll, you’re, you’ve, your, yours, yourself, yourselves
z	Zero

Table 3 shows the number of terms before stop word removal and after stop word removal.

Table 3. Number of Terms before and after Stop word removal

KEYWORDS / NUMBER OF DOCUMENTS	NUMBER OF TERMS EXTRACTED			NUMBER OF TERMS AFTER STOPWORD REMOVAL		
	50	100	200	50	100	200
APPLE	905	1787	2626	756	1338	1997
MYSQL	452	854	1565	377	688	1233
FREE ANTIVIRUS	492	763	1291	408	613	1024
BRIDGE	470	921	1951	397	790	1670
CLUSTER	803	1233	2272	679	1017	1780
DICTIONARY	1134	1469	2229	982	1232	1833
PROCESSOR	483	979	2257	407	809	1760
NIMCET	532	819	1451	423	642	1142
MOUSE	921	1515	2445	766	1237	1973
TAMIL MP3	948	1407	2137	843	1234	1847
MOBILE	630	875	1392	461	646	1088
CAMERA	457	847	1828	358	673	1448
HARDWARE	397	684	1303	346	578	1701
GRID COMPUTING	433	653	1394	344	513	1106
NETWORKING	581	929	1780	487	767	1404
ORANGE	653	1490	3019	566	1266	2434
DATASTRUCTURES	363	763	1349	288	592	1028
DOCOMO	553	1089	1940	460	879	1525
CLOUD COMPUTING	381	613	1145	312	516	927
CAT	645	1298	2274	533	1063	1865

Once the stop words are removed from the collection of extracted terms, stemming is performed to convert the terms into their base forms. Table 4 shows the stemmed terms produced.

Table 4. Stemmed terms

comput, jobs, wholesal, latest, processors, cen, person, wikipedia, free, encyclopediamediawiki, alpha, wmf, tom, new, test, reviewsen, index, follow, tomshardwar, review, internet, premier, resourc, imag, result, hardware, googl, comprehens, search, web, network, cisco, hp, force, f, equip, support, vtqgxb, yokbeisw, zppvmeiocllu, alru, indian, brass, export, door, india, wrought, kavali, manufactur, builder, artwar, iron, knocker, stopper, knob, letter, plate, pull, handl, cabinet, fit, casement, stay, hook, bathroom, mongery, window, fasten, hing, cabin, lifter, security, lock, distributor, hardwar, numer, shutter, solid, centr, chain, bell, push, finger, indiamart, amp, diy, supplier, build, luxuwhviwbfotmiarv, xnx, njttrh, rbakxcva, trader, produc, construct, materi, tool, home, suppli, compon, peripher, electr, electro, mechan, trade, directory, product, global, marketplac, choos, verifi, light, nehru, place, hub, delhi, price, azad, singh, onlin, dealer, community, asia, biggest, market, directly, updat, e, shop, nehruplac, best, cheapest, bazaar, ithub, bb, bc, pc, lcd, monitor, memory, card, ram, hard, disk, drive, pen, dvd, combo, writer, webcam, digit, camera, mp, player, usb, devic, laptop, spare, accessori, batteri, extern, case, printer, classifi, import, adaptor, bluetooth, busi, reader, cabl, cd, duplic, server, replic, servic, cdrw, cpu, fan, dat, tablet, agp, pci, vga, dot, matix, encod, firewir, gpr, modem, hdd, id, laser, mobil, ribbon, scanner, webhost, wireless, lan, zip, intern, hous, exterior, interior, set, signag, switch, plug, cover, coat, hat, curtain, tie, electrophorat, plant, aluminum, forg, stoper, victorian, georgian, item, quality, microsoft, page, open, ccna, certification, job, technic, thousand, industry, naukri, apply, bangalor, mumbai, hyderabad, kolkata, chennai, pune, citi, career, site, softwar, account, time, bank, financ, center, document, day, gener, cach, creer, post, resum, develop, direct, richard, stanley, dylan, mcdermott, stacey, travi, john, lynch, visit, imdb, photo, showtim, cast, crew, plot, summary, comment, discuss, taglin, trailer, poster, messag, board, user, rate, synopsi, credit, book, hwbindex, mediawiki, main, adapt, circuit, connector, consol, inform, yahoo, directorylist, devot, includ, articl, tutori, overclock, idc, compatibility, list, googl, gt, slashdot, nerd, stuff, matter, specif, olpcmediawiki, olpc, translat, w, battery, power, dcon, display, ec, especificacion, etoy, pcquest, hardwarethi, entir, gamut, cut, edg, technolog, launch, applic, stori, focu, fast, pace, track, evolut, linux, os, code, program, pcq, annual, magazin, uncompl, complic, secret,

Once the useful terms are extracted, TFIDF matrix is formed. Table 5 shows the sample TFIDF matrix.

Table 5. TFIDF Matrix

Docs/ Terms	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
D1	0.2	0	0.8	0.3	0.1	0.1	0.4	0	0.1	0.1
D2	0.5	0	0	0	0	0	0	0.1	0.1	0.1
D3	0.2	0	0	0	0	0	0	0	0	0
D4	0.1	0	0	0	0.5	0	0.3	0	0	0
D5	0.8	0	0	0	0	0	0	0	0	0
D6	0.6	0	0.7	0.3	0	0	0	0	0	0

From the resultant TFIDF matrix, distance between every two document is identified using Euclidean function and filled in the distance matrix. Table 6 shows the distance matrix formed.

Table 6. Distance Matrix

Docs	D1	D2	D3	D4	D5	D6
D1	0.0	5.42	1.2	4.3	4.3	16.0
D2	5.42	0.0	4.96	1.8	1.4	21.3
D3	1.2	4.96	0.0	3.49	3.6	17.5
D4	4.3	1.8	3.49	0.0	20.0	34.8
D5	4.3	1.4	3.6	20.0	0.0	20.58
D6	16.0	21.3	17.5	34.8	20.58	0.0

Using the distance matrix, weight of each link is identified and filled in the weight matrix. Table 7 shows the weight matrix formed.

Table 7. Weight Matrix

Docs	D1	D2	D3	D4	D5	D6
D1	1.0	35.29	179.8	44.5	44.5	11.35
D2	35.29	1.0	44.0	177.0	177.56	8.53
D3	179.8	44.0	1.0	59.73	59.46	10. 63
D4	44.5	177 0	59.73	1.0	8.95	5. 64
D5	44.5	177.56	59.46	8.95	1.0	8.62
D6	11.35	8.53	10.63	5.64	8.62	1.0

Using the weight values, the rank of a page is calculated by adding these weights with the existing PageRank formula.

Relevancy of the proposed method is evaluated using the measures precision, recall and F-Measure. Precision and Recall are defined in terms of a set of retrieved documents and a set of relevant documents. Precision is defined as the number of relevant documents retrieved by a search divided by the total number of documents retrieved by that search, and recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents. Every result retrieved by a search was relevant if a precision is 1 and all relevant documents are retrieved by the search if a recall is 1. F-measure is computed by combining the values of precision and recall.

Precision is calculated as,

$$\text{Precision} = \frac{\left|\{\text{Relevant Documents}\} \cap \{\text{Retrieved Documents}\}\right|}{\left|\{\text{Retrieved Documents}\}\right|} \quad (18)$$

Recall is calculated as,

$$\text{Recall} = \frac{\left|\{\text{Relevant Documents}\} \cap \{\text{Retrieved Documents}\}\right|}{\left|\{\text{Relevant Documents}\}\right|} \quad (19)$$

F-Measure is calculated as,

$$\text{F - Measure} = 2\text{X}\ \frac{\text{Precision}\ \ \text{X Recall}}{\text{Precisio}\ \ + \text{Recall}} \quad (20)$$

Proposed technique is compared with existing algorithms by considering different page sizes and observed that the proposed work is producing more f-measure when compared to existing ranking algorithms.

Figure 1 and Figure 2 show that the precision and recall of the proposed method are better than the existing methods.

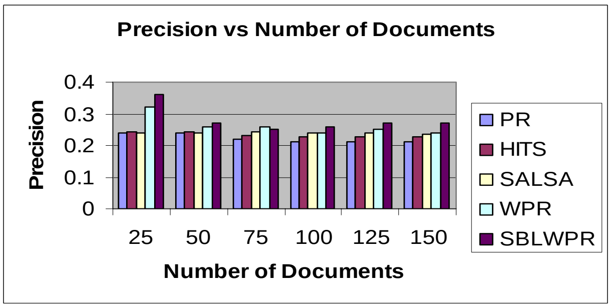


Fig. 1 – Precision Vs Number of Documents

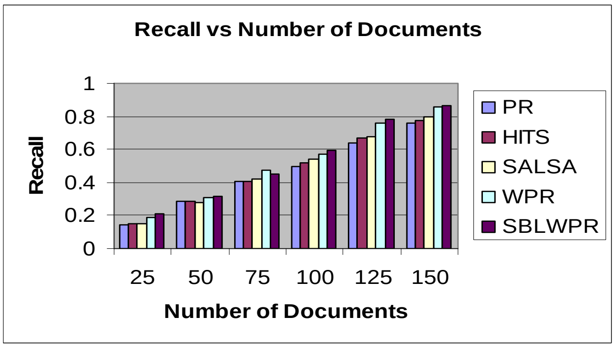


Fig. 2 – Recall Vs Number of Documents

Figure 3 show that the proposed method is gaining more F-Measure than other ranking algorithms such as PageRank, HITS, SALSA and Weighted PageRank at different page sizes.

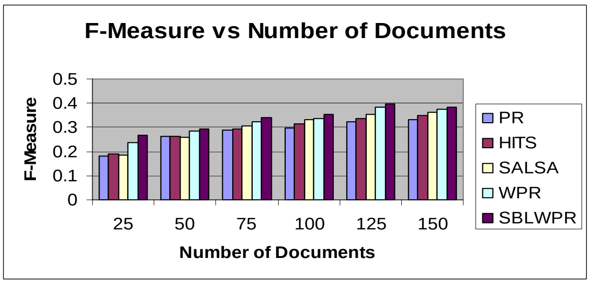


Fig. 3 – F-measure Vs Number of Documents

5. CONCLUSION

In this paper, a new method is proposed to order the search results based on the similarity among the linked documents. In the existing ranking algorithms, rank of a page is equally distributed to the pages to which it has a link and no weight is assigned to the links by looking at the similarity among the linked documents. Normally, a page is more important if it has more number of inlinks from the documents which are similar to it. Using this concept as a base, we introduced a method to assign weight to each link. Results show that the proposed method makes more relevant documents to appear at the beginning of a list of returned results when compared to other existing methods.

6. FUTURE WORK

This algorithm calculates similarity among the linked documents in addition to PageRank value. Due to this, it takes much time to compute the rank of the pages. This computational time can be reduced by optimizing the algorithm. Avoiding web content spamming can be taken as a future work since the performance of the proposed method may be affected by introducing identical documents.

7. REFERENCES

[1] R. Kosala and H. Blockeel, Web mining research, *A survey, ACM SIGKDD Explorations*, 2 (1) (2000). pp. 1-15.

[2] S. Madria, S. S. Bhowmick, W. K. Ng, and E.-P. Lim, Research issues in web data mining, *Proceedings of the Conference on Data Warehousing and Knowledge Discovery*, (1999). – pp. 303-319.

[3] S. Pal, V. Talwar, and P. Mitra, Web mining in soft computing framework: relevance, state of the art and future directions, *IEEE Transactions on Neural Networks*, 13 (5) (2002). – pp. 1163-1177.

[4] S. Brin and L. Page, The anatomy of a large-scale hypertextual web search engine, *Computer Networks and ISDN Systems*, 30 (1-7) (1998). – pp. 107-117.

[5] J. Kleinberg, Authoritative sources in a hyperlinked environment, *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, (1998). – pp. 668-677.

[6] Ask Jeeves, Inc., *Teoma search engine*, http://www.teoma.com.

[7] S. Chakrabarti, B. Dom, D. Gibson, J. Klein–berg, P. Raghavan, and S. Rajagopalan, Automatic resource list compilation by analyzing hyperlink structure and associated text, *Proceedings of the 7th International World Wide Web Conference*, (1998). – pp. 65-74.

[8] R. Lempel and S. Moran, The stochastic approach for link-structure analysis (SALSA) and the TKC effect, *ACM Transactions on Information Systems*, 19 (2000). – pp. 387-401.

[9] D. Zhang and Y. Dong, An efficient algorithm to rank web resources, Computer Networks: *The International Journal of Computer and Telecommunications networking*, 33 (1-6) (2000). – pp. 449-455.

[10] A. N. Langville and Carl D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press, 2006.

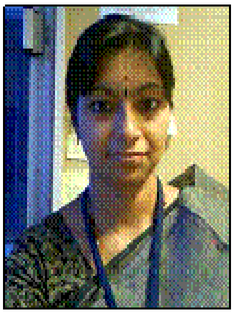
[11] R. Lempel and S. Moran, SALSA: The Stochastic Approach for Link-Structure analysis, *ACM Transactions on Information Systems*, 19(2) (2001). pp. 131-160.

[12] Wenpu Xing and Ali Ghorbani, Weighted PageRank algorithm, *Second Annual Conference on Communication Networks and Services Research* (CNSR'04), (2004). – pp. 305-314.

[13] S. Chakrabarti, B. Dom, and P. Indyk, Enhanced hypertext categorization using hyperlinks, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (1998). – pp. 307-318.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Technical report Stanford Digital Libraries, SIDL-WP-1999-0120, 1999.

Ms S. Poomagal, *obtained her B.Sc (Applied Science – Computer Technology) from Bharathiar University, Coimbatore in the year 2001. She completed her M.Sc (Applied Science – Computer Technology) from Bharathiar University, Coimbatore in the year 2003. She completed her M.Phil (Computer Science) from Bharathiar University in the year 2004. She is pursuing her Ph.D (Information Technology) in Anna University, Coimbatore. She worked as a lecturer in Dr. SNS Rajalakshmi College of Arts & Science, Coimbatore from Jan 2005 to April 2007. She is working as an assistant professor in the Department of Computer and Information Sciences, PSG College of Technology, Coimbatore since June 2007. She has published 5 papers in various national conferences and international journals. Her research interests include Data mining, Compiler design and Programming Languages.*



Dr. T. Hamsapriya, *obtained her BE (Electronics and Communication Engineering) from Bharathiar University, Coimbatore in the year 1988. She completed her ME (Communication systems) from Bharathiar University, Coimbatore in the year 1992. She completed her Ph.D (Parallel Computing) from Anna University, Chennai in the year 2008. She worked in Kumaraguru College of Technology, Coimbatore as an associate lecturer from 1993-94. She worked in BPL Electronics as a consultant engineer from 1994-96. She was acting as a managing partner in Computer Software College fro 1996-99. She is working as a professor in PSG College of Technology, Coimbatore since 1999. She is the head of the Department of Information Technology, PSG College of Technology, Coimbatore. She is also a research cordinator in PSG College of Technology. She has published more than 25 papers in various international conferences and journals.*