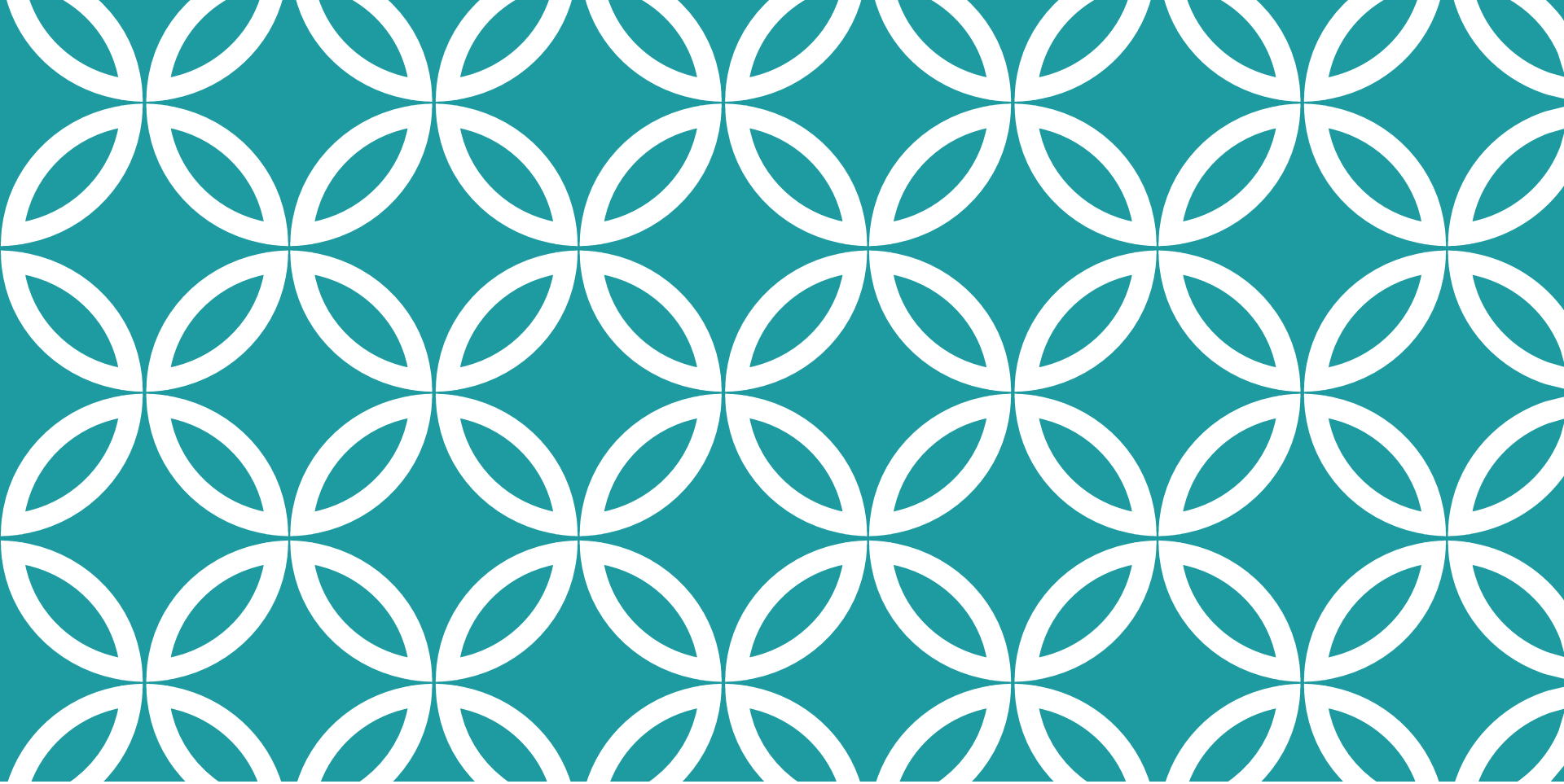


ORGANIC WEB SEARCH:

Page Rank Algorithm –

Finding Authoritative Web Pages

CT102 Information Systems

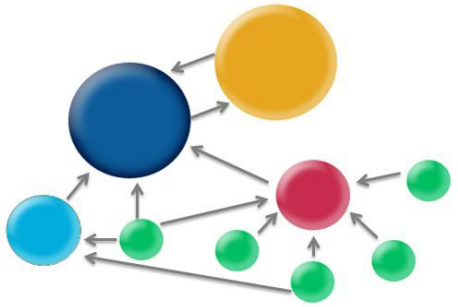
SERP (Search Engine Results PageS) and Web Page Ranking

Ranking involves ordering the results returned in response to a user query by one or more scores which are assigned to web pages by the search engine.

These scores can be assigned based on:

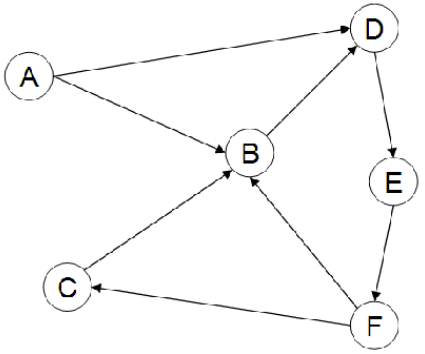
- Similarity scores (i.e. Euclidean dot product similarity)
- Potentially Page rank scores (organic – using links-pre-calculated)
- Ad word scores (sponsored - at search time)
- Personalised Information (Language, Location, Previous web searches, Browsing history, Tracking information, etc.)
- Featured articles/Question Answering/Knowledge Graph information

PAGE RANK



- Page rank uses a link analysis algorithm to determine the *relative* importance of each web page.
- Page rank was originally developed by the founders of Google and was one factor which led to Google's dominance initially.
- The approach is currently used in Google and other search engines.
- Page rank only considers the in and out (hypertext) links of a web page (not the content, and not internal links).
- Each web page can be assigned a **PageRank** score.

OVERVIEW



- Idea: The higher the page rank score the more important the page and such pages should be ranked above pages with lower page rank scores (if both pages are returned in response to a query and are otherwise equally relevant to the query)
- The general idea is that a link from a web page is a "vote" or "endorsement" of the page it links to.
- The Page Rank algorithm tries to "sum up" all the votes for pages.
- However each page that "votes" also has its own Page Rank score and *spreads* its vote over all the pages it links to so that a "vote" from an important page is more important than a "vote" from a less important page.
- Uses a link analysis algorithm to determine the *relative* importance of each web page

The original Page Rank paper

http://infolab.stanford.edu/~backrub/google.html

The Anatomy of a Large-Scale Hypertextual Web Search Engine

Sergey Brin and Lawrence Page
{sergey, page}@cs.stanford.edu
Computer Science Department, Stanford University, Stanford, CA 94305

Abstract

In this paper, we present Google, a prototype of a large-scale search engine which makes heavy use of the structure present in hypertext. Google is designed to crawl and index the Web efficiently and produce much more satisfying search results than existing systems. The prototype with a full text and hyperlink database of at least 24 million pages is available at http://google.stanford.edu/

To engineer a search engine is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. Despite the importance of large-scale search engines on the web, very little academic research has been done on them. Furthermore, due to rapid advance in technology and web proliferation, creating a web search engine today is very different from three years ago. This paper provides an in-depth description of our large-scale web search engine -- the first such detailed public description we know of to date.

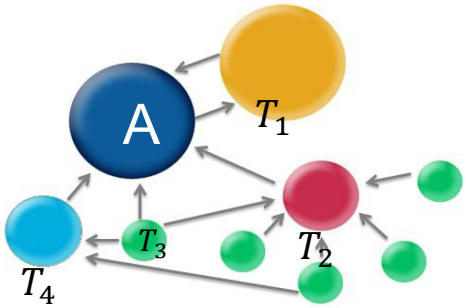
Apart from the problems of scaling traditional search techniques to data of this magnitude, there are new technical challenges involved with using the additional information present in hypertext to produce better search results. This paper addresses this question of how to build a practical large-scale system which can exploit the additional information present in hypertext. Also we look at the problem of how to effectively deal with uncontrolled hypertext collections where anyone can publish anything they want.

Keywords: World Wide Web, Search Engines, Information Retrieval, PageRank, Google

1. Introduction

(Note: There are two versions of this paper -- a longer full version and a shorter printed version. The full version is available on the web and the conference CD-ROM.)
The web creates new challenges for information retrieval. The amount of information on the web is growing rapidly, as well as the number of new users inexperienced in the art of web research. People are likely to surf the web using its link graph, often starting with high quality human maintained indices such as Yahoo! or with search engines. Human maintained lists cover popular topics effectively but are subjective, expensive to build and maintain, slow to improve, and cannot cover all esoteric topics. Automated search engines that rely on keyword matching usually return too many low quality matches. To make matters worse, some advertisers attempt to gain people's attention by taking measures meant to mislead automated search engines. We have built a large-scale search engine which addresses many of the problems of existing systems. It makes especially heavy use of the additional structure present in hypertext to provide much higher quality search results. We chose our system name, Google, because it is a common spelling of googol, or 10^{100} and fits well with our goal of building very large-scale search engines.

More simply …

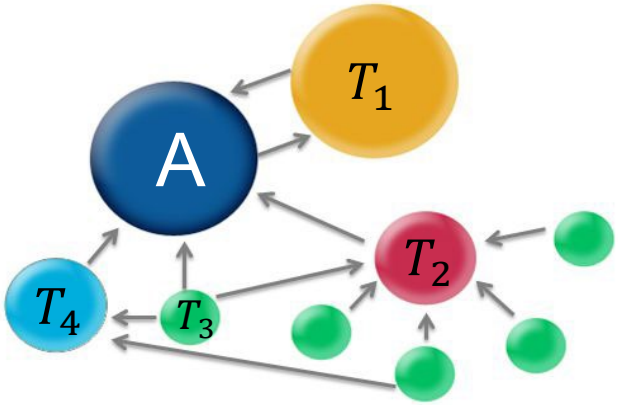


If A is a web page, as shown in the figure, linked to by 4 other web pages, T_1, T_2 , T_3 , T_4 and C(T) is the total number of links from a web page T (outlinks) then the Page Rank of A, for N pages PR(A) is =

$$(1\text{-}d)/N + d * (PR(T_1)/C(T_1) + PR(T_2)/C(T_2) + PR(T_3)/C(T_3) + PR(T_4)/C(T_4))$$

N.
B.

General Formula…



If:

A is a web page, as shown in the figure, linked to by 4 other web pages, T_1, T_2 , T_3 , T_4 with N = 9 web pages in total

C(T) is the total number of (out)links from any web page T

Then the Page Rank of A, for N pages PR(A) is =

$(1\text{-}d)/N + d * (PR(T_1)/C(T_1) + PR(T_2)/C(T_2) + PR(T_3)/C(T_3) + PR(T_4)/C(T_4))$

In general for any web page A with web pages T_1 … T_R linking to it

… and also knowing for each web page, C(T), the total number of (out)links from a web page T and N the total number of web pages:

The Page Rank of A is defined as:

PR(A) =

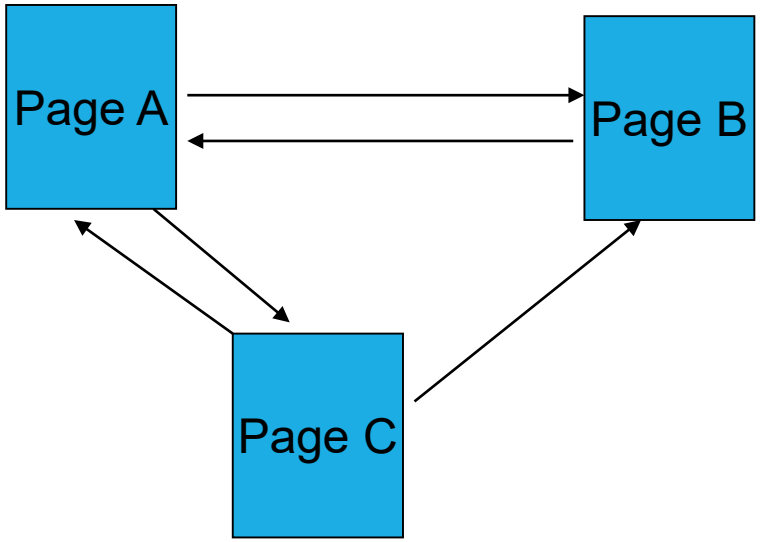
$(1\text{-}d)/N + d * (PR(T_1)/C(T_1) + \ldots + PR(T_R)/C(T_R))$

i.e., summing up, for each page linking to A, the ratio of the "Page Rank" value coming from each page

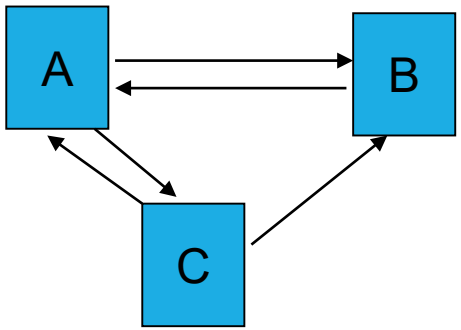
To get started ….

- We do not know the page rank of the pages to use – this needs to be calculated.
- Each page must be given an initial (estimate/guess) of a Page Rank score (usually 1/N) - this value is initially used as a guess for all PR values needed (all pages)
- This is modified over a number of iterations until it *converges* (settles) to some value, i.e. it only changes by a very small number, such as .001, (if at all), from one iteration to the next
- For each iteration, the PR scores of each page from the previous iteration is used in the calculations
- d is usually set to 0.85 (this has been proven to speed up convergence)

EXAMPLE 1 Calculate the page rank scores for each of the 3 web pages in the following graph. Take d = 0.85

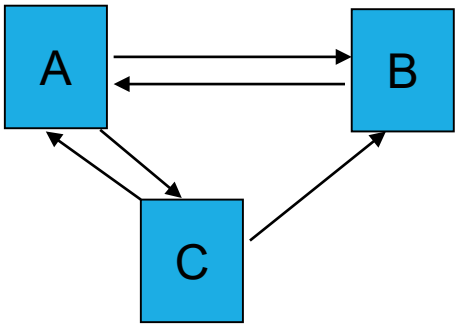


INITIAL STEPS:



- Count C() for each page: C(A) = 2, C(B) = 1, C(C) = 2
- For web page A: vote comes from B and C
- For web page B: vote comes from A and C
- For web page C: vote comes from A only
- Assign initial PRs: PR(A)=PR(B)=PR(C)= 1/3
- Set d = 0.85
- Write the formula for each page:
- PR(page) = $(1\text{-}d)/N + d\ (PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$
- Keep calculating the formula for each page using the scores from previous iterations until convergence is reached

Write the formula for each page and start calculating:



- PR(page) = *(1-d)/N + d (PR(T1)/C(T1) + … + PR(Tn)/C(Tn))*
- Store the PR result from the previous calculation in *prevA, prevB, prevC*
- prA = 0.15/3 + 0.85 * (prevB/cB + prevC/cC);
- prB = 0.15/3 + 0.85 * (prevA/cA + prevC/cC);
- prC = 0.15/3 + 0.85 * (prevA/cA);
- prevA = prA;
- prevB = prB;
- prevC = prC;

WRITING IN C CODE

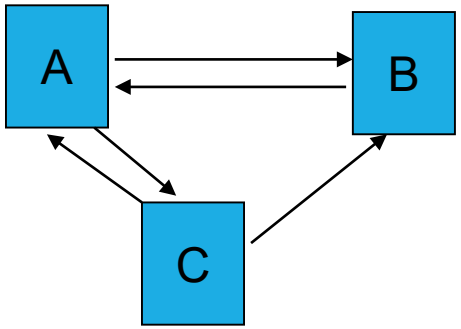
```
prA = prB =prC = 1.0/3.0;
prevA = prevB = prevC = 0.0;
cA = 2;
cB = 1;
cC = 2;
for(i=0; i < 10; i++)
 {
   prevA = prA;
   prevB = prB;
   prevC = prC;
   prA = 0.15/3 + 0.85 * (prevB/cB + prevC/cC);
   prB = 0.15/3 + 0.85 * (prevA/cA + prevC/cC);
   prC = 0.15/3 + 0.85 * (prevA/cA);
}
```

CALCULATE IN EXCEL OR EQUIVALENT:

C(A)	C(B)	C(C)
2	1	2
PR(A)	**PR(B)**	**PR(C)**
0.15/3 + 0.85*(PR(B)/C(B) + PR(C)/C(C))	0.15/3 + 0.85*(PR(A)/C(A) + PR(C)/C(C))	0.15/3 + 0.85*(PR(A)/C(A))
0.333333333	0.333333	0.333333333
0.475	0.333333	0.191666667
0.414791667	0.333333	0.251875
0.440380208	0.333333	0.226286458
0.429505078	0.333333	0.237161589
0.434127008	0.333333	0.232539658
0.432162688	0.333333	0.234503979
0.432997524	0.333333	0.233669142
0.432642719	0.333333	0.234023948
0.432793511	0.333333	0.233873156
0.432729424	0.333333	0.233937242

RESULTS (to 4 decimal places):

After 10 iterations:



PR(A) = 0.4327

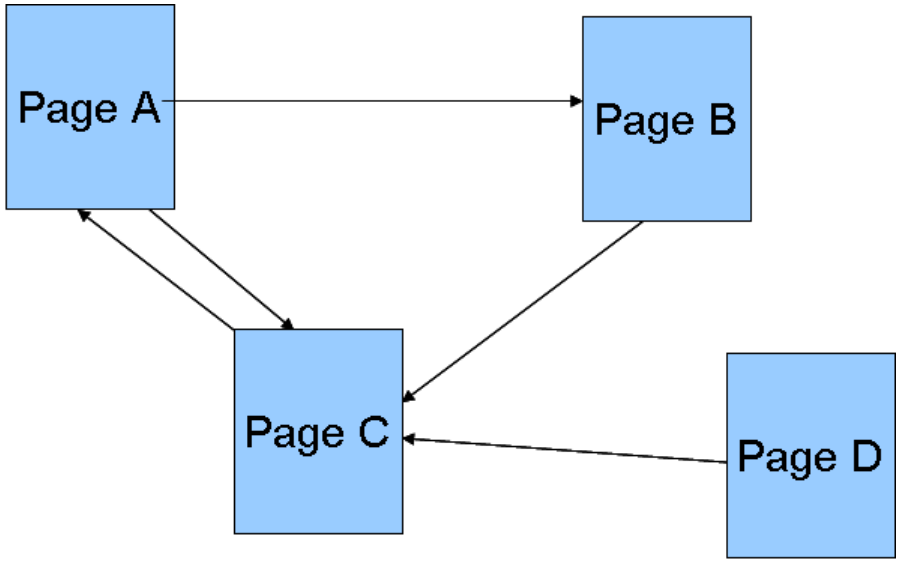
PR(B) = 0.3333

PR(C) = 0.2339

How many iterations are required?

- Usually compare the difference between the last two values of the PR scores per web page (e.g. prA and prevA)
- When this difference is small, e.g., .001 can stop
- In general the number of iterations will depend on the number of web pages
- For a small number of web pages can set the number of iterations (e.g., 10 in previous example)
- For web indexed pages (millions of web pages), can converge in less than 100 iterations

YOU TRY: EXAMPLE 2: Calculate the page rank scores for each of the 4 web pages in the following graph. Take $d = 0.85$



Steps:

1. Specify the inlinks and the number of outlinks for each webpage

2. Write the formula for each web page

3. Code or use Excel to get results to 4 decimal places

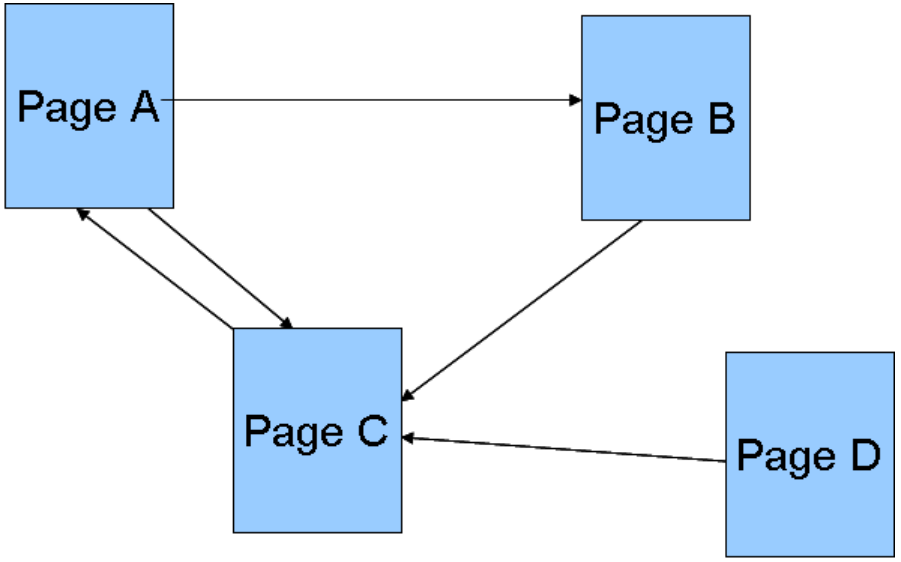
Answer after 14 iterations
(to 4 decimal places)

PR(A) = 0.3722

PR(B) = 0.1959

PR(C) = 0.3944

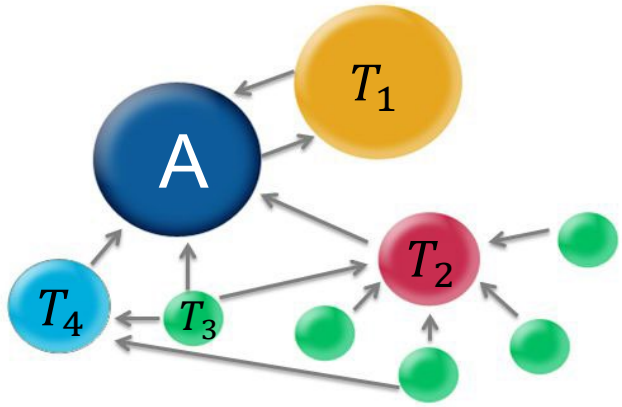
PR(D) = 0.0375



NOTES:

- According to the Page Rank formula, page D gets a Page Rank value (of 0.0375) even though it has no links to it
- In reality pages with no links will be discarded from the calculation at the start and just given the 1-d/N score at the start

YOU TRY: Original example: Can you calculate the page rank scores for each of the 9 web pages in the original graph? Take d = 0.85



RANKING *based on* COSINE SIMILARITY SCORE *and* PAGE RANK SCORE

The (cosine) similarity score is the most important measure for organic search

Only web pages whose similarity scores are above a certain threshold value would be considered for display on the first page of the search engine's result page.

In deciding between web pages which have the same of very similar similarities, the page rank values of these pages would be used such that a page with a higher page rank score would be displayed above/instead of a page with lower page rank score.

For example:

sim(webpage1, query) = 0.2511.. and PageRank(webpage1) = 0.351

sim(webpage2, query) = 0.2511… and PageRank(webpage2) = 0.151

=> webpage1 displayed above/instead of webpage2

SUMMARY OF PAGE RANK

The page rank approach is a common algorithm used by search engines to find *authorative* or more important (organic) web pages. It uses the web link structure – and no content.

It can be seen as a measure of *crowdsourced quality* of a page relative to other pages.

It is independent of any query and any personalisation.

It can be pre-calculated based on the information gathered by the crawler and stored in the index (i.e., a PR score is associated with each page in the index).

It only needs to be updated if there is a change in the link structure (i.e. inlinks/outlinks) of web pages.

SUMMARY OF WEB SEARCH

- Representing natural languages by means of vectors of real-valued numbers is a fundamental abstraction in Web Search … there are many ways to obtain these vectors .. we have considered one approach
- Euclidean dot product and Page Rank scores are two very important calculations in Web Search
- These concepts and measures also have many other applications in related areas, especially with respect to Natural
  - Any application where we have unstructured text (natural language) we write programs which will represent the text by real-valued vectors
  - Any application where we can represent "things" by real valued vectors, we can find the similarity between those vectors using Euclidean dot product
  - Anywhere we have directed connections between "things" (i.e. in and out web links) we can find the "crowdsourced" most authorative/important things.