Data Mining and Web Algorithms

- Why web search algorithms(information Retrieval systems)
- Different Retrieval Systems
  - Boolean Retrieval
  - Ranked Retrieval
  - Link Analysis
  - Ranking algorithms
- Web Crawlers
- Web Caching Algorithms
- Recommendation Systems

Recap

Original PageRank algorithm :-

→ It is an iterative algorithm that starts with given.

$$PR(A) = \underbrace{(1-d)} + d\left[\frac{PR(T_1)}{C(T_1)} + \frac{PR(T_2)}{C(T_2)} \cdots \frac{PR(T_n)}{C(T_n)}\right]$$

where, $\boxed{\text{the probability for random surfer jumping to a page is always } (1-d).}$

→ $PR(A)$ = PageRank of page A

→ $PR(T_i)$ = PageRank of Pages T_i which link to page A (inbounds link)

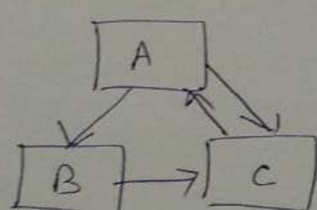
→ $C(T_i)$ = No of outbound links on page T_i

→ d = damping factor lies B/w 0 & 1.

→ According to Page and Brin (who have proposed)

$d = 0.85$.

Eg



No of Nodes = 3

Let us assume initial are 1.

$$PR(A) = (1-d) + d\left[\frac{PR(C)}{C(C)}\right]$$

$$= (0.15) + (0.85)\left[\frac{PR(C)}{1}\right] \quad \text{—— (1)}$$

$$PR(B) = (1-d) + d\left[\frac{PR(A)}{C(A)}\right]$$

$$= (1-0.85) + (0.85)\left[\frac{PR(A)}{2}\right] \quad \text{—— (2)}$$

(7)

$$PR(C) = (0.5) + (0.5)\left[\frac{PR(A)}{C(A)} + \frac{PR(B)}{C(B)}\right]$$

$$= 0.5 + 0.5\left[\frac{PR(A)}{2} + \frac{PR(B)}{1}\right] \quad \text{--- (3)}$$

Let

$\boxed{PR(A) = P(B) = P(C) = 1}$ initial $\boxed{\text{0th iteration}}$

$PR(A) = 0.5 + 0.5 = 1$

$PR(B) = 0.75$

$PR(C) = 0.5 + 0.5[0.5 + 0.75]$

$= 0.5 + 0.5 * 1.25$

$= 0.5 + \cancel{75}.625 = 1.125$

$\boxed{\text{Ist iteration}}$

Sum of PageRank of all the pages = No. of web pages

$= 1 + .75 + 1.125 = 2.875 \approx 3$

It has to repeated (iteratively) until we get the same (or almost same) value of Page Rank.

Another notation of pagerank n° :-

$$PR(A) = \frac{(1-d)}{N} + d\left[\frac{PR(T1)}{C(T1)} + \frac{PR(T2)}{C(T2)} \cdots \frac{PR(Tn)}{C(Tn)}\right]$$

Then sum of all pages will be One.

(*)

$$PR(C) = (0.5) + (0.5)\left[\frac{PR(A)}{C(A)} + \frac{PR(B)}{C(B)}\right]$$

$$= 0.5 + 0.5\left[\frac{PR(A)}{2} + \frac{PR(B)}{1}\right] \quad \text{--- (3)}$$

Let

$\boxed{PR(A) = P(B) = P(C) = 1}$ initial $\boxed{0\text{th iteration}}$

$PR(A) = 0.5 + 0.5 = 1$

$PR(B) = 0.75$

$PR(C) = 0.5 + 0.5[0.5 + .75]$

$= 0.5 + 0.5 * 1.25$

$= 0.5 + \cancel{.75}.625 = 1.125$

$\boxed{I^{st}\text{ iteration}}$

Sum of PageRank of all the pages = No. of web pages

$= 1 + .75 + 1.125 = 2.875 \approx 3$

It has to repeated (iteratively) until we get the same (or almost same) value of page Rank.

Another notation of pagerank n° :-

$$PR(A) = \frac{(1-d)}{N} + d\left[\frac{PR(T1)}{C(T1)} + \frac{PR(T2)}{C(T2)} \cdots \frac{PR(Tn)}{C(Tn)}\right]$$

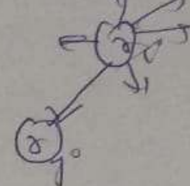
Then sum of all pages will be one.

⑧

Determining the pagerank with above equations becomes difficult if we have numerous equations in large web. [Secondly, it assumes that every node has non-zero outdegree]

Power Iteration Method :-

Consider web as a graph (nodes are pages and edges are hyperlinks :-



Now the earlier equation :-

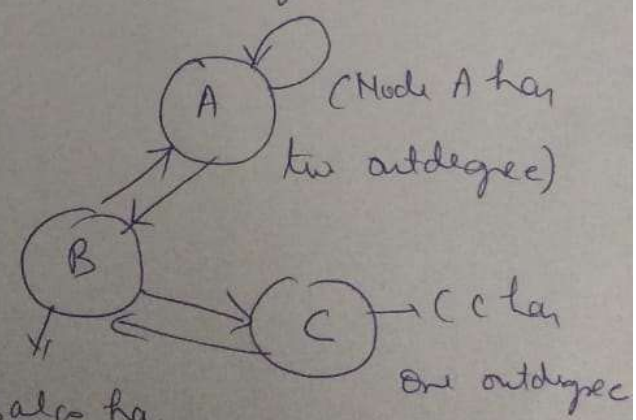
$$\boxed{r_j^{(t+1)} = \sum_{i \to j} \frac{r_i^{(t)}}{d_i}}$$

(Page Rank of Page j at $(t+1)$th iteration) Can be written in the matrix form as

$$r^{(t+1)} = M \cdot r^{(t)}$$

Here, M is the matrix of the order $N \times N$ if N is No of nodes in the webgraph.

$$M = \begin{array}{c|ccc} & A & B & C \\ \hline A & \frac{1}{2} & \frac{1}{2} & 0 \\ B & \frac{1}{2} & 0 & 1 \\ C & 0 & \frac{1}{2} & 0 \end{array}$$



(4) Algo :-

$\rightarrow$ or $r^0 = [\frac{1}{N}, \frac{1}{N}, \frac{1}{N}, \cdots \frac{1}{N}]^T$ (Transfer)

(1) Initialize: $r^0 = \left(\frac{1}{N}\right)$ for all the pages

(2) Iterate: $r^{(t+1)} = M \cdot r^{(t)}$

(3) Stop: when $|r^{(t+1)} - r^{(t)}| \leq \epsilon$ or these ranks are same for last two iterations.

Repeated

$$r^0 = [1/3, 1/3, 1/3]^T = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$r^{t+1} = M \cdot r^t$$

$\rightarrow$ M is column matrix

$\rightarrow$ it is stochastic i.e. sum of all column entries is 1 in each column

$$r^1 = \underset{3\times 3}{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}} \underset{3\times 1}{\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}}$$

$$= \begin{bmatrix} \frac{1}{2}\times\frac{1}{3} + \frac{1}{2}\times\frac{1}{3} \\ \frac{1}{2}\times\frac{1}{3} + \frac{1}{3} \\ 0 + \frac{1}{2}\times\frac{1}{3} + 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ \frac{3}{6} \\ \frac{1}{6} \end{bmatrix}$$

$$r^2 = M \cdot r^1$$

$$= \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{3}{6} \\ 1/6 \end{bmatrix} = \begin{bmatrix} \frac{1}{6} + \frac{1}{4} + 0 \\ \frac{1}{6} + 0 + \frac{1}{6} \\ \frac{1}{2} \times \frac{\cancel{3}}{\cancel{6}2} \end{bmatrix}$$

$$= \begin{bmatrix} 5/12 \\ \frac{1}{3} \\ \frac{1}{4} \end{bmatrix}$$

After some iteration it will converge.

(10)

In case of random surfer model, damping factor | teleporting factor d or β or α is given then:-

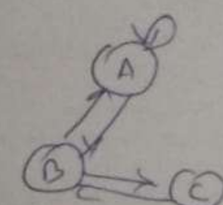
matrix equation become

$$\boxed{r^{(t+1)} = (\alpha \cdot M)(r^{t})}$$

typically, $\alpha = 0.8 \,|\, 0.9$ [or in Between]

Example:-

$$M = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \end{array}$$



$$r^{0} = \left[\frac{1}{3} \;\; \frac{1}{3} \;\; \frac{1}{3}\right]^{T}$$

$$r^{1} = 0.8 \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$$

$$= \begin{bmatrix} .4 & .4 & 0 \\ .4 & 0 & .8 \\ 0 & .4 & 0 \end{bmatrix} \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} = \begin{bmatrix} .8/3 \\ 1.2/3 \\ .4/3 \end{bmatrix}$$

Do it iteratively to get the result (or convergence condition is met.

(11)

Problems :-

① "Spider Trap" Problem (~~Sink Problem~~)

Eg 1 

→ Outgoing links are in the group. i.e. in above example A can not go to any other page and vice versa.

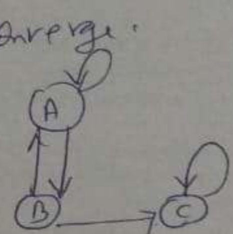
if we initialize the page rank with

$r_a = 1 \;, 0 \;, 1 \;, 0$

$r_b = 0 \;, 1 \;, 0 \;, 1 \; ---$

It will never converge.

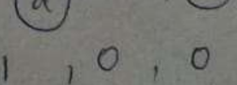
Eg 2: C is a spider trap as it cannot go anywhere else except C.



→ This problem can be solved with teleporting.

② "Dead end" Problem (Dangling Nodes | No outgoing links)

→ It happens when a node has no outgoing link. Then its importantance can not be transferred.



$r_A = 1 \;, 0 \;, 0 \;, 0$
$r_B = 0 \;, 1 \;, 0 \;, 0$ } Score is zero after some iterations

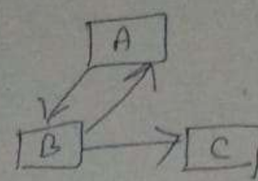
→ This problem can be solved by replacing zero entries with $\frac{1}{N}$

(12)

Eg

	A	B	C
A	0	1/2	0
B	1	0	0
C	0	1/2	0

$= M$



So remove the dead end problem by replacing 3rd column with $\frac{1}{N}$ i.e. $\frac{1}{3}$.

$$\begin{array}{c c} & \begin{array}{ccc} A & B & C \end{array} \\ \begin{array}{c} A \\ B \\ C \end{array} & \begin{bmatrix} 0 & 1/2 & 1/3 \\ 1 & 0 & 1/3 \\ 0 & 1/2 & 1/3 \end{bmatrix} \end{array} = M$$

Google Matrix equation is :-

$$\boxed{A = \beta . M + (1-\beta)\left(\frac{1}{N}\right)_{N \times N}}$$

then calculate rank iteratively by following equation :-

$$\boxed{r^{t+1} = A . r^{t}}$$

β = damping / teleporting factor

M = Matrix (column-stochastic matrix)

A = Google Matrix

Reference : www.mmds.org. (Stanford University)

Mining of Massive datasets : Leskovec, Rajaraman, & Ullman.

Hubs and Authorities

HITS (Hyperlink - Induced Topic Search)

→ It is developed by John Kleinberg in 1997.

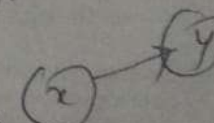
→ Every web page is assigned two score, one is called Hub score and other is its authority Score

→ For any query, we compute two ranked lists i.e. hub score list & authority score list

→ A good hub page is one which points to many good authorities & a good authority page is one which is pointed to by many good hub pages.

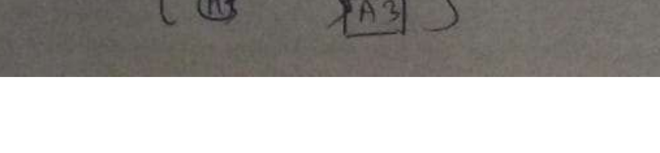
→ Let us take a subset of webgraph and then compute its hub & authority score.

for a webpage x, x has a hyperlink that points to y.



Hub score of node $x = h(x) = \sum_{x \to y} a(y)$

authority score of node $x = a(x) = \sum_{y \to x} h(y)$



Algorithm :-

1. Assign each node an authority and hub score of webgraph.

   i.e. initially $h(x) = a(x) = 1$ or $h(x) = a(x) = \frac{1}{\sqrt{M}}$

2. Apply the authority update rule: each node's authority score is the sum of

   indegree

   hub scores of each node that it points to.

   $$a(x) = \sum_{y \to x} h(y)$$

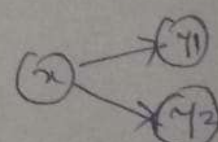
   

3. Apply the hub update rule: each node's hub score is sum of authority score of

   outdegree

   each node that it points to

   $$h(x) = \sum_{x \to y} a(y)$$

4. Normalize authority and hub scores:

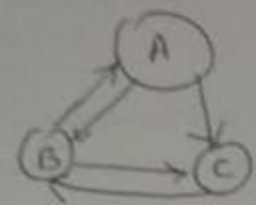
   $$auth(j) = \frac{auth(j)}{\sum_{i \in M} auth(i)}$$

   $$hub(j) = \frac{hub(j)}{\sum_{i \in M} hub(i)}$$

5. Repeat 2, 3, 4 until convergence condition is met i.e. scores become constant.

Example :-

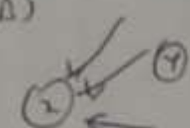
Step 1
Let $a_i^0 = h_i^0 = 1$



Step 2

	old A	old H	New A	New H
A	1	1	1 (B)	2 (B & C)
B	1	1	2 (A & C)	2 (A & C)
C	1	1	2 (A & B)	1 (A)

$$\text{authority (indegree)}\ (x) = \sum_{y \to x} h(y)$$



$$\text{hub}(x) = \sum_{x \to y} a(y)$$

Normalize :- $\text{authority}(A) = \dfrac{A(A)}{A(A)+A(B)+A(C)} = \dfrac{1}{5}$

Similarly for other nodes

	new A	new H	new A	new H
A	1/5	2/5	H(B) 2/5	A(B)+A(C) 4/5
B	2/5	2/5	H(A)+H(C) 3/5	A(A)+A(C) 3/5
C	2/5	1/5	H(A)+H(B) 4/5	A(B) 2/5

and now normalize and do it iteratively till stopping condition is met

Matrix Notation :-

Let vector $a = (a_1, a_2 \ldots a_n)$ be a authority vector

$h = (h_1, h_2 \ldots h_n)$ be a hub vector

n = no of nodes in the web graph.

Step1: initialize $a_i = h_i = \frac{1}{\sqrt{n}}$ or

$a_i = h_i = 1$

Step2: Repeat until stopping condition is met (or it converges)

(i) $h = A \cdot a \Rightarrow h = (AA^T) \cdot h$

(ii) $a = A^T \cdot h \Rightarrow a = (A^T A) \cdot a$

(iii) Normalize a & h. (Use L_1 or L_2 norm)

Step3:- If convergence criterion is met, stop else go to step2.

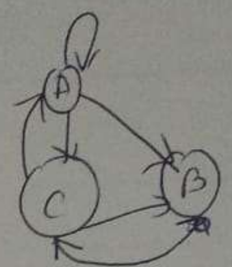
L_2 Norm is:- $\sum_{i \in N} (a_i)^2 = 1$ & $\sum_{i \in N} (h_i)^2 = 1$

Hence, at every iteration. normalize each value of a & h by

$$a_i^{(t)} = \frac{a_i^{(t)}}{\sqrt{\sum_{j=1}^{n} a_j^2}} \text{ and } h_i^{(t)} = \frac{h_i^{(t)}}{\sqrt{\sum_{j=1}^{n} h_j^2}}$$

$$a_i^{(t)} = \frac{a_i^{(t)}}{\sqrt{a_1^2 + a_2^2 + \ldots a_n^2}} \text{ and } h_i^{(t)} = \frac{h_i^{(t)}}{\sqrt{h_1^2 + h_2^2 \ldots h_n^2}}$$

Example :-
$$[A] = \begin{array}{c c} & \begin{array}{ccc} A & B & C \end{array} \\ \begin{array}{c} A \\ B \\ C \end{array} & \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$$



$$a = (A^T \times A \cdot a)$$
$$h = (A \times A^T \cdot h)$$

$$A^T A = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

$$A\,A^T = \begin{bmatrix} 3 & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 2 \end{bmatrix}$$

$$a = [A^T A] \cdot a$$
$$= \begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 4 \end{bmatrix}$$

$$h = [AA^T] h = \begin{bmatrix} 3 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 4 \end{bmatrix}$$

next iteration

$$a = \begin{bmatrix} 24 \\ 24 \\ 18 \end{bmatrix}$$

$$h = \begin{bmatrix} 28 \\ 8 \\ 20 \end{bmatrix}$$

and so on till it converges.

①

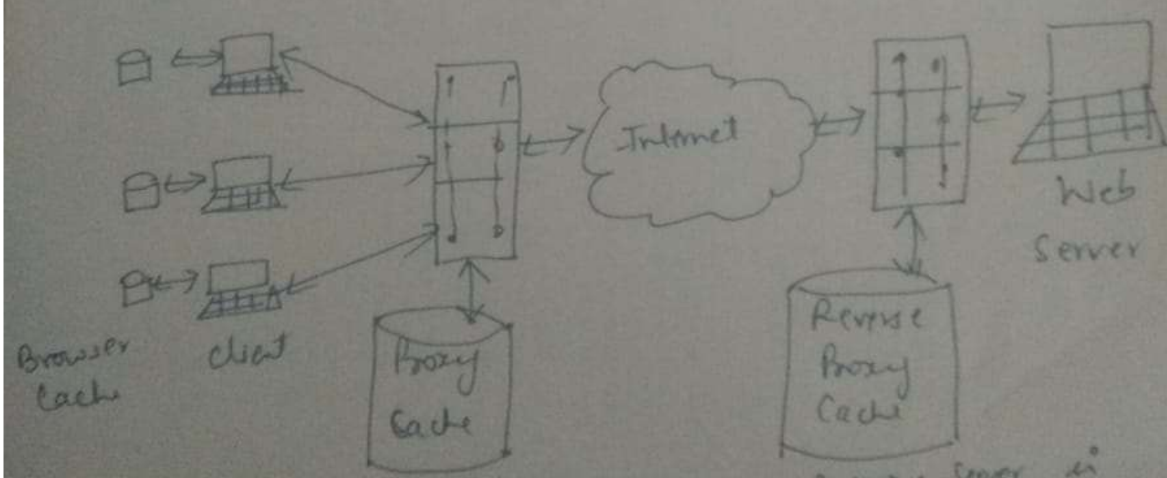
World Wide Web Cacheing (www)

→ Web Caching (www) provides an efficient remedy to the latency problem.

→ It brings the documents closer to clients in less latency (time).

→ Caching can be deployed at various points in the internet Eg

- client browser
- Proxy server | Reverse proxy server



Proxy Cache :- It is kept in proxy servers. A proxy server is a computer that is often placed near a gateway to the internet (as shown in figure) and provides shared cache to the clients.

→ Cache mechanism help to reduce network and server load by minimizing transmission of documents which are on demand.

→ However, every cache is naturally limited by its size. Whenever the cache is full and the proxy needs to ~~...~~ cache a new document, then cache replacement policies have to be incorporated.

Cache replacement policies tries to increase the hit ratio of cache. Here is some parameters defined:-

(i) Cache hit : when data is found in cache.

(ii) Cache Miss :- When data is not found in cache.

(iii) Hit Ratio (Hit rate) :- % of times, data is found in the cache.

(iv) Miss ratio (Miss Rate) :- % of times, data is not found in the cache

$$\boxed{\text{Miss rate} = 1 - \text{hit rate (hit ratio)}}$$

(v) Miss Penality :- time to move data from server to cache.

→ Normally, caching's popular types are application caches and memory caches for their ability to speed up certain responses.

→ Web caching is the core design feature of HTTP protocol to minimize traffic while improving the responsiveness of the system.

What can be cached:-

→ Logos and brand images
→ Non-Rotating images (navigation icons)
→ style sheet
→ Java Script files
→ Media files
→ Downloadable content

What not to be cached:-

→ Assets related to Sensitive data
→ Content which is user-specific and frequently changed

Book:- Web Caching by Duane Wessels
Publisher: O, Relly Media, Inc. (3)

Cache Replacement :-

→ It refers to the process that takes place when cache becomes full and old objects must be removed to make space for new ones.

→ Now, question is, which has to be removed?

→ To answer this question, cache researchers and developers have proposed different replacement algorithms discussed below:-

(i) LRU (Least Recently Used)

→ Most popular replacement algorithm used by web caches.

→ It removes the objects that have not been accessed for the longest time.

→ LRU would consider time-since-reference as the only parameter.

→ In practice, web caches almost always use a variant called LRU-threshold. The "threshold" refers to object size. Objects size larger than the threashold size are not cached.

LRV, FIFO, LRU, Random, OPT

(u)

→ Caching numerous small objects results in a higher hit ratio

(ii) FIFO (first in, first out)

→ It is simple to implement

→ Objects ~~that~~ are purged ~~[illegible]~~ in the same order they added.

→ It is not popular because it gives lower hit ratio than LRU.

(iii) LRV (Lowest Relative Value)

It expels the object that has "lowest utility value". In LRV, the utility of a document is calculated adaptively on the basis of data readily available to proxy server. It performs better than LRU. To compute the utility, it uses cost, size and last reference time. ~~to calculate~~. The cost of a document includes the time and processing overhead associated with retrieving the document from original / proxy server.

(IV) Optimal Algorithm (OPT)

⇒ The document/page that will not be used for the longest period of time in the future is replaced. It involves the knowledge of future requests to predict which item in the cache will be needed again.

⇒ It has lowest miss rate, but it is difficult to implement.

(V) Randomized Algorithms :-

⇒ These algorithms expels a document drawn randomly from the cache.

⇒ It means, it uses the ~~concept~~ concept of probability (Concept of randomness)

⇒ Example:-

RANDOM :-

→ It randomly selects M documents from the cache.

→ Select the least useful document among them for replacement based on the given policy (existing algo like LRU, FIFO etc.)

→ The next M least useful documents out of M are kept for the next iteration.

→ Iterations are continued until there is enough space in the cache to accomodate the remaining documents.

Web Crawler

Web search engine (web- IR) can be visualized in following figure [1]. In Web, getting the content of the documents takes longer time because of latency, Web search engines must crawl their documents. A web crawler does following operations:

- Initialize queue with URLs of known seed pages
- Repeat
  - Take URL from queue
  - Fetch and parse page
  - Extract URLs from page
  - Add URLs to queue

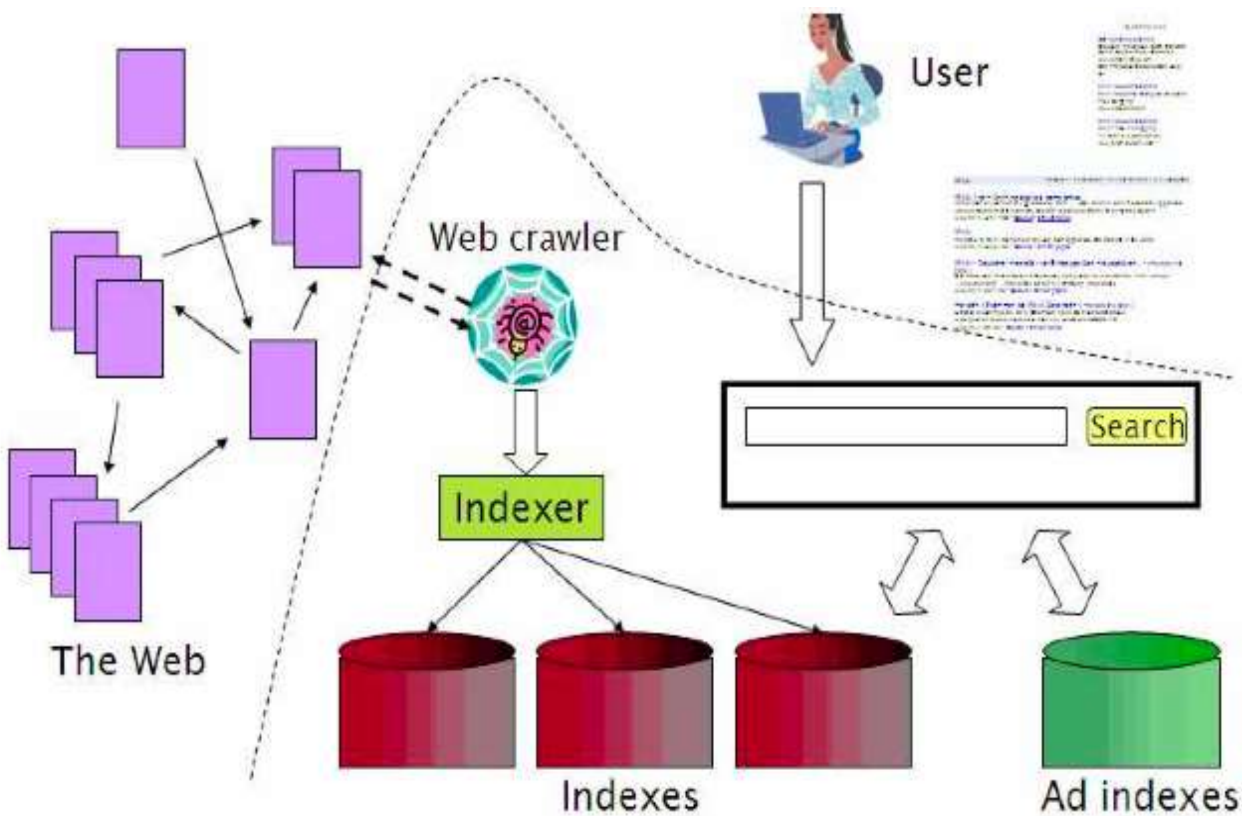


Figure 1: The various components of a web search engine

Web Crawler Architecture

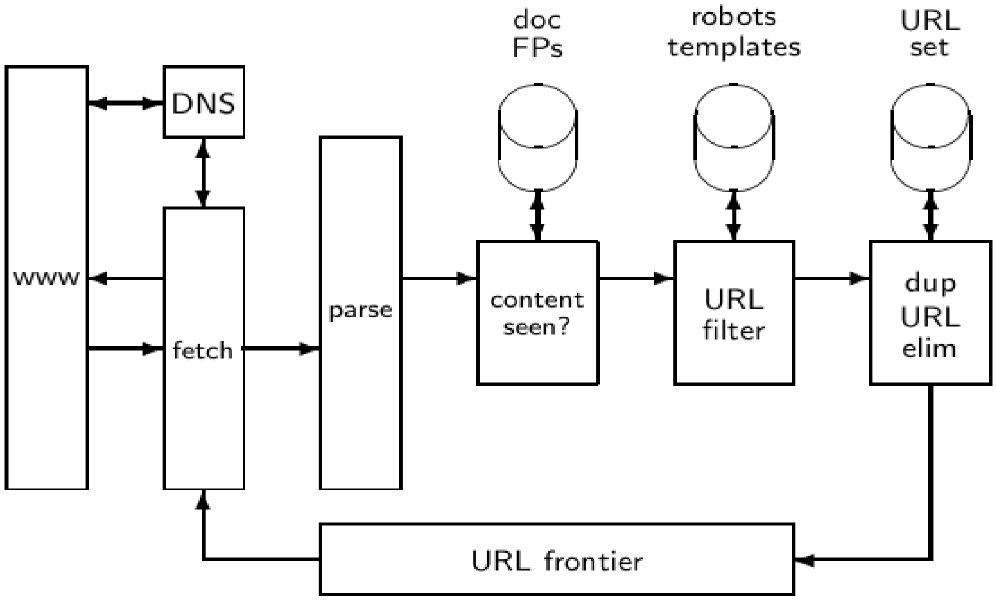


Figure 2: Web Crawler Architecture

<u>Processing steps of crawler:</u>

1. Pick a URL from the frontier
2. Fetch the document at the URL
3. Parse the URL
    a. Extract links from it to other docs (URLs)
4. Check if URL has content already seen
    a. If not, add to indexes
5. For each extracted URL
    a. Ensure it passes certain URL filter tests
    b. Check if it is already in the frontier (duplicate URL elimination)

References:

[1] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. Cambridge university press, 2008.

[2] Leskovec, Jure, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive data sets. Cambridge university press, 2020.

[3] https://www.mmds.org

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

[5] Wessels, Duane. Web caching. " O'Reilly Media, Inc.", 2001.

[6] Cao, Pei, and Sandy Irani. "Cost-aware www proxy caching algorithms." Usenix symposium on internet technologies and systems. Vol. 12. No. 97. 1997.

[7] Balamash, Abdullah, and Marwan Krunz. "An overview of web caching replacement algorithms." IEEE Communications Surveys & Tutorials 6.2 (2004): 44-56.