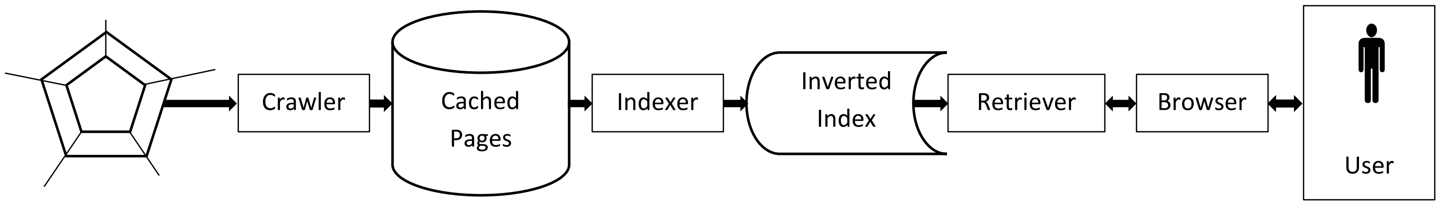
Text Retrieval and Search Engines

Basic Web Search



Crawler

A **crawler**, also known as a **spider** or a **robot**, gathers information from the web and stores it in a data cache for further analysis and information retrieval. A simple approach to building a crawler is to seed it with a small number of starting pages and have it fetch and parse those pages for further hyperlinks which are then searched in the same way. In practice a number of complications need to be taken into account to handle the complexities of the web. These include varying file types, dynamically generated pages and so on.

There are a number of strategies which can be adopted to search efficiently and accurately including breadth-first searching of linked documents and parallel crawling approaches. Focused crawling can also be adopted for more specific topics. Often all these strategies need to include an iterative component to maximize information retrieval.

Breadth First Search

A breadth first search traverses a tree structure by scanning the neighboring nodes before moving onto the next level neighbors.

Google File System

A distributed file system which allows the data to be store on multiple machines.

MapReduce

A framework for producing the index of the cached data in parallel.

Hadoop

An open source implementation of MapReduce.

Ranking Web Search Results

The standard information retrieval algorithms apply to web searches but extensions to the standard are required in a web search. Sometimes a web search represents a navigation query where a user is seeking an entry point to a tree of linked documents pertaining to a subject, rather than the single document with the most relevance. Additionally web documents have added dimensions such as **links** or **number of click-throughs** to be considered. Usually many different ranking mechanisms are combined by machine learning algorithms.

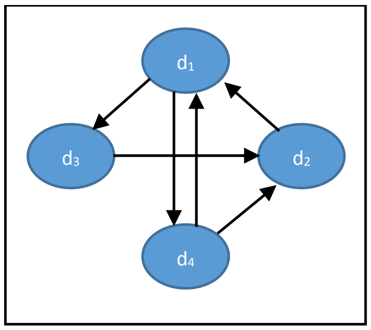
Page Rank

Page rank is a way of determining the significance of a page based on the number of links to it. It also takes into account the page rank of the linking pages, thus a page linked from another page which itself has a lot of links would be classed as having a high page rank.

Page rank is usually smoothed so that all pages are considered to have a non-zero number of links to them. This allows the page rank to be determined using linear algebra techniques

The PageRank Algorithm

One approach to solving the page ranking problem is to calculate the probability α that a user will randomly jump to another page. The probability of picking a linked page to follow is (1 - α). A simple way to calculate this is to count the number of links on the page and generate a **Transition Matrix** *M*:



$$M = \begin{bmatrix} 0 & 0 & 1/2 & 1/2 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \end{bmatrix}$$

$$M_{ij} = probability\ of\ going\ from\ d_i\ to\ d_j$$

$$\sum_{j=1}^{N} M_{ij} = 1$$

Thus the probability of visiting a page d_j at time $t+1$ is:

$$p_{t+1}(d_j) = (1-\alpha)\sum_{i=1}^{N} M_{ij}p_t(d_i) + \alpha\sum_{i=1}^{N}\frac{1}{N}p_t(d_i)$$

Since we are only interested in the probability of the next page at any given time we can drop the time index and the probability equation becomes:

$$p(d_j) = \sum_{i=1}^{N}\left[\frac{1}{N}\alpha + (1-\alpha)M_{ij}\right]p(d_i)$$

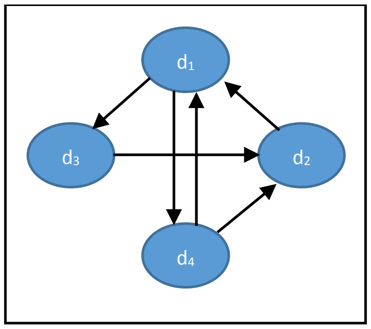
Or the following (which can be solved iteratively):

$$\bar{p} = (\alpha I + (1-\alpha)M)^T\bar{p}\ \ where\ I_{ij} = 1/N$$

Hypertext-Induced Topic Search (HITS) for Page Ranking

HITS is another approach to solving the page ranking problem, it assumes that pages that are widely cited are good **authorities** and pages that cite many other pages are good **hubs** thus there is an iterative reinforcement of their rank.

In order to analyze the relationship between authorities and hubs we calculate the Adjacency Matrix *A*.



$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$Inital\ values\ a(d_i) = a(d_i) = 1$$

$$\left.\begin{aligned} h(d_i) &= \sum_{d_j \in OUT(d_i)} a(d_j) \\ a(d_i) &= \sum_{d_j \in IN(d_i)} h(d_j) \end{aligned}\right\}\ iterate$$

$$Normalize: \sum_i a(d_i)^2 = \sum_i h(d_i)^2 = 1$$

$$\bar{h} = A\bar{a};\ \ \bar{a} = A^T\bar{h}$$
$$\bar{h} = AA^T\bar{h};\ \ \bar{a} = A^TA\bar{a}$$

Combining Features in Ranking

For a given query document pair *(Q, D)* we can determine various features (BM25 score, page rank etc.). It is possible to generate a more powerful ranking function by combining all the features in one ranking algorithm using the hypothesis:

$$p(R|Q,D) = s(X_i(Q,D),\dots,X_n(Q,D),\lambda)$$

λ represents a set of parameters which we can learn by fitting the function with training data, commonly in the form of 3-tuples *(D, Q, 1)* for relevant data and *(D, Q, 0)*for non-relevant data. This fitting is usually done using machine learning techniques.

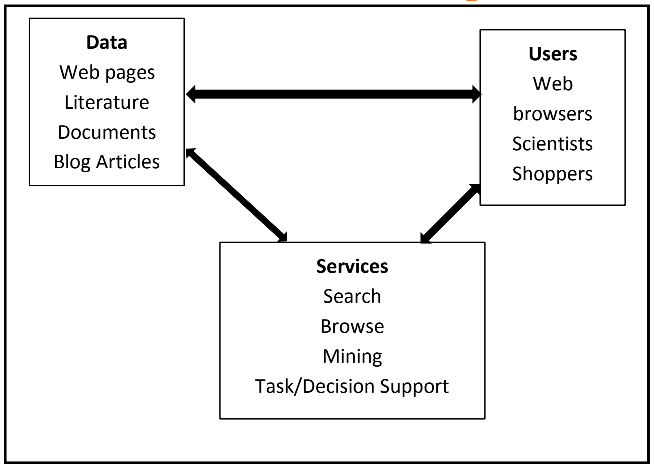
Combining Features in Ranking Using Regression

$$log\left(\frac{P(R=1|Q,D)}{1-P(R=1|Q,D)}\right) = \beta_o + \sum_{i=1}^{n} \beta_i X_i$$

Where β_0 and β_1 are parameters to be determined from the training data and we use the log value to ensure that the calculation remains with the range 0 to 1. This becomes:

$$P(R=1|Q,D) = \frac{1}{1+exp(-\beta_o - \sum_{i=1}^{n} \beta_i X_i)}$$

Data-User-Service Triangle



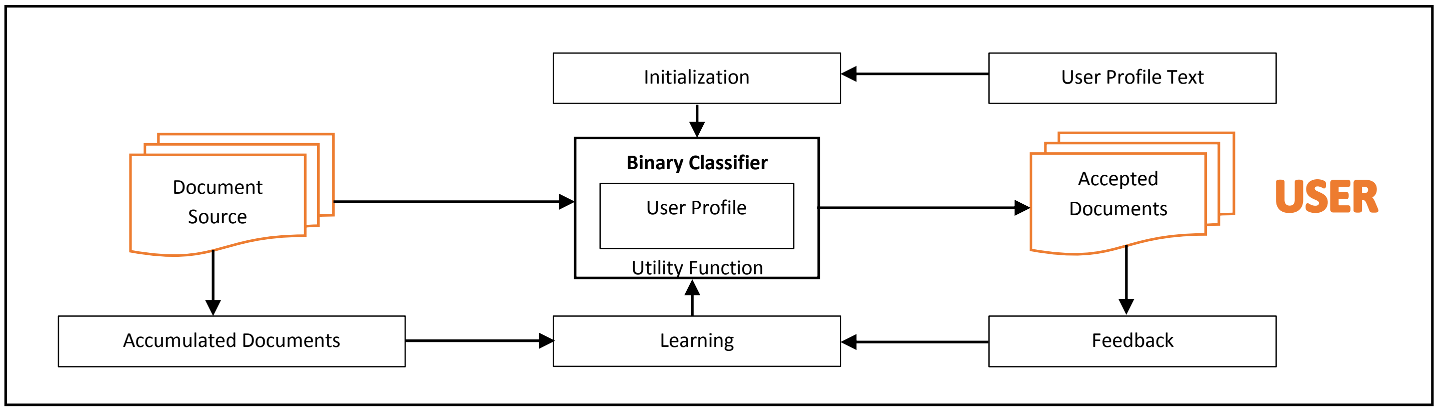
The data-user-service triangle refers to the flow of information from data sources to users and services.

Each of the three nodes of the triangle are being extensively researched in order to get more useful results. The “Users” area is being extended to model users in order to make search and query more personalized and relevant. The “Data” section is undergoing increased processing in order to further understand the underlying meaning behind the text in the documents to provide a true knowledge representation of the data. The “Services” section is incorporating more powerful mining and analysis to give stronger, more interactive task support to query problems.

Recommender Systems

Recommender Systems or **Filter Systems** represent information retrieval in **push mode** (as opposed to the pull mode query search techniques discussed previously). There are two approaches to building a recommender system. One is to look at items the user has previously selected and look for similar items. Item similarity based filtering is known as **Content-Based Filtering**. The second approach is to examine what type of user likes item X and then check if the current user is similar. User similarity based filtering is known as **Collaborative-Filtering**. These two approaches can be combined to increase their efficacy.

Typical Content-Based Filtering System



This system is based on standard text retrieval techniques using a **score threshold θ** for a filtering decision. The problem then becomes one of setting the threshold and checking the utility of the system. Adaptive learning mechanisms are used to determine the appropriate value for the threshold.

Linear Utility

One way to quantify the utility of the content-based filtering system is to count the number of relevant or "good" documents and compare with the number of non-relevant or "bad" documents:

$$Linear\ Utility = \left(c_g N_{good}\right) - \left(c_b N_{bad}\right)$$

The constants c_g and c_b can be adjusted to vary the amount of exploration the system will accommodate when carrying out information retrieval.

Beta-Gamma Threshold Learning

A set of training documents can be used to calculate the optimal score threshold at which maximum utility of the system is achieved but it may be that documents with a score below this threshold are still of interest. The beta-gamma threshold is a heuristic approach to lowering the threshold based on a number of configurable parameters. The threshold of a system can be expressed as:

$$\theta = \alpha\theta_{zero} + (1-\alpha)\theta_{optimal}$$

θ_{zero} is the threshold where the system utility value reaches zero.
$\theta_{optimal}$ is the threshold that gives the maximum utility value of the system.
$\alpha = \beta + (1-\beta)e^{-N\gamma}$ where N is the number of training examples.
β helps prevent over fitting to the training data.
γ controls the influence of the number of samples in the training data set, the more examples in the training set the closer θ becomes to $\theta_{optimal}$.

Collaborative Filtering (CF)

Collaborative filtering is based on user similarity, it assumes that a user *u* will have the same preferences as the set of similar users $\{u_1, \dots, u_m\}$. If the rank of an object o_i by user u_j is X_{ij} then the problem becomes one of determining the function $f(u, o) \rightarrow X$ from the training data.

Memory-Based Approach to Collaborative Filtering

If X_{ij} is the rating of an object o_j by user u_i and n_i is the average rating of all objects by user u_i then we can normalize the ratings to $V_{ij} = X_{ij} - n_j$

The prediction of the rating of object o_j by user u_a then becomes:

$$\hat{v}_{aj} = k\sum_{i=1}^{m} w(a,i)v_{ij}$$

Where $k = \frac{1}{\sum_{i=1}^{m} w(a,i)}$

So $\hat{x}_{aj} = \hat{v}_{aj} + n_a$

Where $w(a,i)$ is a weighting function that weights the ratings more heavily in favor of users who are similar to u_a and *k* is a normalizing factor. More specifically $w(a,i)$ measures the similarity between user u_a and u_i so the problem becomes one of calculating $w(a,i)$. It should be noted that this approach does not deal with the problem of missing values or Inverse User Frequency which is similar to the Inverse Document Frequency problem of text retrieval.

Examples of User Similarity Measures

The following are two of the many different measures used to determine the value of $w(a,i)$ used in collaborative filtering systems.

The Pearson Correlation Coefficient	The Cosine Measure
$w_p(a,i) = \frac{\sum_j (x_{aj}-n_a)(x_{ij}-n_i)}{\sqrt{\sum_j (x_{aj}-n_a)^2 \sum_j (x_{ij}-n_i)^2}}$	$w_c(a,i) = \frac{\sum_j x_{aj}x_{ij}}{\sqrt{\sum_j {x_{aj}}^2 \sum_j {x_{ij}}^2}}$