

# Knowledge discovery in the Internet

Piotr Gawrysiak, Michał Okoniewski

Institute of Computer Science, Warsaw University of Technology  
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland  
{gawrysia, okoniews}@ii.pw.edu.pl

**Abstract:** With the rapid expansion of the World Wide Web, the need for efficient data retrieval strategies becomes stronger and will be still growing. Unfortunately classical information retrieval techniques, developed for well-organized collections of textual data do not seem to be able to cope with diversity and amount of information available throughout the Internet. This paper presents some of the newest approaches to information retrieval in large, unstructured hypertext spaces - such as WWW - that focus more on latent information embedded in hyperlinks and document structure, then on actual understanding of Web pages textual content. These techniques, that are marking the new trends and prospects for the Internet technology, have been given recently the name "Web mining", as in fact they are examples of unsupervised machine learning similar to data mining and text mining. Here we discuss methods belonging to the following three groups: link topology analysis, statistical text analysis and query languages and systems design.

**Keywords:** webmining, text mining, information retrieval, search engines, Internet

## 1. Introduction and motivation

Almost everyone agrees that current state of the art in the Internet search engine technology renders extracting information from Web a skill not many people are able to master. Widely used search engines, such as [W1] and [W2] are plagued either by the lack of precision or by inadequate recall rate. They tend to return thousands of answers for even specific queries while from time to time refusing to find appropriate documents albeit they exist and are accessible through the net.

Almost all commercial search engines use classical keyword-based methods for information retrieval. That means that they try to match user specified pattern (i.e. query) to texts of all documents in their database, returning these documents that contain terms from the query.

Such methods are quite effective for well-controlled collections - such as bibliographic CD-ROMs or handcrafted scientific information repositories. Unfortunately the Internet has not been created, but it rather evolved and therefore cannot be treated as well controlled collection. It contains a lot of irrelevant and redundant information and what is maybe even more important - it does not have any kind of underlying semantic structure, that could facilitate navigation.

Some of the above issues are result of improper query construction. The questions directed to search engines are often too general (like "water sources" or "capitals") and this produces millions of returned documents. The texts that the user was interested in are probably among them, but cannot be separated as the human attention has its limitations. One hundred documents is generally regarded as maximum amount of information that a human mind can interpret and treat as useful in such situations.

On the other hand documents sometimes can not be retrieved because the specified pattern was not matched exactly. This can be caused by flexion in some languages, or by confusion introduced by synonyms and complex idiom structures. For example English word Mike is often given as an example of this, as it can be used as a male name or a shortened form of a noun "microphone".

Most search engines also have very poor user interfaces. The computer aided query construction systems are very rare, and search result presentation concentrates mostly on individual documents, not allowing for more general overview of retrieved data (which could be very important when number of returned documents is huge).

The simple remedy for above problems simply does not exist, yet recent advances in data mining and automatic knowledge discovery fields give hope that similar methods could be applied also to data structures accessible through the Internet. Indeed, we try to present in this paper some of the new breed of information retrieval techniques that work well in highly chaotic Internet environment. This ability is made possible by using not only the content of documents (be it text, images or even binary files) but also by exploiting information that serves humans who are browsing Web. It occurs that information contained in hyperlink topology, formatting, or even in the patterns of language usage can be helpful for automated information retrieval system. Meanwhile innovative user interfaces and query languages give us better opportunity to express our retrieval goals, and to assess the information that has been extracted.

The goal of this paper is to present and assess the functionality of various Web mining methods. Presumably, many of them do affect or will influence commercial Internet applications. The methods presented in this paper are divided into two major groups described in separate sections. The first one emphasises the research effort on Web topology and structural dependencies between links and documents, without examining contents of hypertext documents. It proves that the structure of Web itself carries lots of useful information that should be retrieved. There are many possible applications of this knowledge, yet it seems that most robust are: (1) ranking page importance using the density of hyperlink network, and (2) grouping pages into heavily hyperlinked clusters. The first approach is the core of Page Rank method, developed by L. Page and S. Brin group at Stanford University. The second methodology has been used by J. Kleinberg to create a system called HITS, which is able not only to cluster pages but also to assign two types of ranking indices (called hub and authority cores) to a web page. Further research on this system resulted in an automatic web resources clustering system called ARC.

We present also hypertext systems that analyse page usage information in order to reconfigure hyperlink structure or enhance search engine heuristics. The methods described include Dynamic Nearness algorithms developed by M. Merzbacher and spreading activation systems of P. Piroli.

These systems and algorithms are described in section 2.

Second group of methods is based on exploring page contents with text mining methods. These techniques use mostly context or text structure analysis. The Latent Semantic Analysis developed by D. Dumais and T. Landauer is a cocurrence analysis method that makes possible searching for synonyms without employing a

thesaurus. The other described method involves analysing formatting structure of the document to extract textual information encapsulated in tables or lists, using manually created "wrapper systems". Descriptions of these approaches have been included in section 3.

Section 4 contains descriptions of advanced query languages designed especially with Internet in mind - such as IW3QL and WebSQL. We present also automatic query factorisation methods, developed by H. Shutze and J. Petersen as a modification of LSA, Finally we describe some visualisation methods such as semantic maps, and hyperbolic trees. These tools usually enhance browsing process and have no direct impact on automatic searching methods, yet can be also used in a query results post processing system.

The final section includes discussion and comments on potential usefulness of these techniques.

## 2. Web topology analysis

### 2.1 Ranking pages using link topology

One of the main problems that plague modern search engines is poor relevance. In many situations, search engines retrieve thousands of pages at least partially satisfying input query. Of course human attention remains more or less constant, and most humans are able to cope only with about 100-200 retrieved documents. We need therefore a method that would allow us to approximate page importance for the user, regardless of its relevance to the query measured by classical methods. Having such approximation - in fact a ranking score - we can either sort the pages, presenting these with higher rank first, or even remove entirely these which have very poor rank - thus allowing user to concentrate on a set of pages which has reasonable quantity.

This is not easy, as without any further clues from the user it is hard to say if a particular page is of *high quality* or *important*. Two methods described below try to extract this very general and uncertain knowledge from the latent information encoded in hyperlink structure of the Internet.

#### PageRank Index

One of the most obvious types of link analysis that has been applied for quite a long time in the area of bibliometrics is citation analysis. Similar methods could be also useful in hypermedia environments.

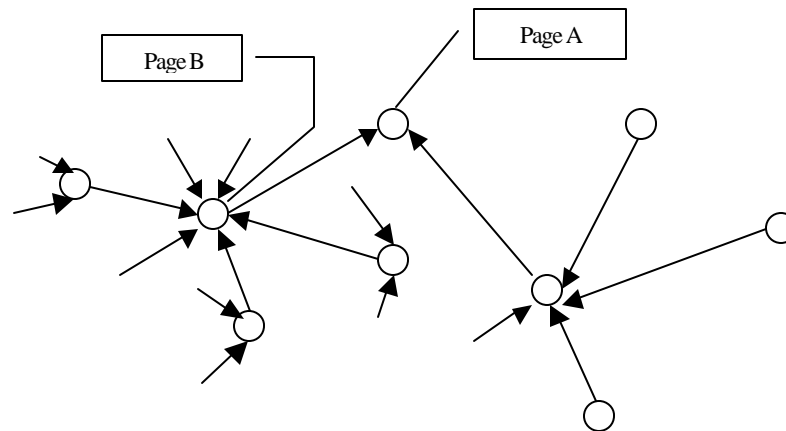
Let  $u$  be an analysed document, and  $B_u$  the set of documents having citations from  $u$  (i.e. connected by *back-links* with  $u$ ),  $F_u$  the set of documents cited by document  $u$  (connected by *out-links* with  $u$ ). We can now define the ranking index  $\mathbf{a}$  of document  $u$  as follows:

$$\mathbf{a}(u) = c|B_u|$$

where  $c$  is a normalization factor, and  $|s|$  denotes cardinality of set  $s$

According to this formula a document should be regarded important and of high quality if many authors include citations from this document in their publications.

If we assume that  $u$  denotes a web page, and the sets  $B_u$  and  $F_u$  incoming and outgoing hyperlinks for this page respectively, we can apply the  $\hat{a}$  index to Internet hypertext pages. However, even a short analysis shows us, that we need some kind of rank propagation mechanism, as many important pages are pointed by relatively few, but highly interconnected documents. Consider for example the following situation:



**Figure 1**

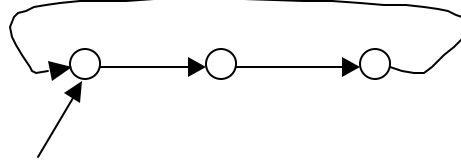
The *Page A* has fewer back-links pointing to it than *Page B*. Ranking index  $\hat{a}$  assigned to it would be therefore much lower, yet we intuitively know, that *Page A* should be regarded important, because is pointed to by highly cited pages. This observation leads to more advanced ranking formula  $\hat{a}$ , as conceived by L. Page [1]:

$$\mathbf{b}(u) = c \sum_{v \in B_u} \frac{\mathbf{b}(v)}{N_v},$$

where  $N_v = |F_v|$  (i.e. the number of out-links of the page  $v$ ).

In other words a page can have high  $\hat{a}$  index, if it is pointed by many pages, or it has relatively few back-links, but coming from pages with high index value.

This index works well in hypertext environments that can be represented by either non-cyclic, or heavily interconnected graphs. Unfortunately the structure of the World Wide Web is different, and the situations as presented below are not uncommon:



**Figure 2**

Such closed link loop has been given a name *rank sink* by Page. Indeed, above pages recursively accumulate rank, and do not distribute it.

To overcome this problem, a slightly modified version of ranking function has been proposed in [1]:

$$\mathbf{d}(u) = c \sum_{v \in B_u} \frac{\mathbf{d}(v)}{N_v} + cE(u),$$

where  $E(u)$  is a constant used to create personalised measure of the rank,  $\|\hat{\mathbf{a}}\| = 1$  and  $c$  is maximal,  $\|\hat{\mathbf{a}}\|$  denotes the norm of the vector  $\hat{\mathbf{a}}$ .

We can represent above indices in a different way. Let  $A$  be a modified coincidence matrix, such that

$A_{u,v} = \frac{1}{N_u}$  if there is a hyperlink from page  $u$  to page  $v$ , and  $A_{u,v} = 0$  otherwise. Let  $\hat{\mathbf{a}}$  and  $\ddot{\mathbf{a}}$  be the vectors

over web pages similar to  $\mathbf{E}$  that contain values of  $\hat{a}$  and  $\ddot{a}$  indices respectively for all web pages. We can now rewrite above equations:

$$\hat{\mathbf{a}} = c\mathbf{A}\hat{\mathbf{a}}$$

$$\ddot{\mathbf{a}} = c(\mathbf{A} + \mathbf{E} \times \mathbf{1})\ddot{\mathbf{a}}$$

Note that  $\hat{\mathbf{a}}$  and  $\ddot{\mathbf{a}}$  are eigenvectors of matrices  $A$  and  $(\mathbf{A} + \mathbf{E} \times \mathbf{1})$  respectively.

The computation of  $\ddot{\mathbf{a}}$  is not a trivial task, as its definition is recursive and at first glance it seems that to find out its value for a given page, we should solve very complex set of equations. We can albeit apply iterative calculation, after initial selection of starting index values:

**S** : starting vector<sup>1</sup>

**E** : rank source vector (usually is uniform, with values set to 0.15)

PageRank( $A, S, E, \hat{a}$ ) :

```
1  R := S ;
2  Repeat
3  Z := AR ;
4   $d := ||\mathbf{R}|| - ||\mathbf{Z}||$  ;
5  Z := Z +  $d\mathbf{E}$  ;
6   $a := ||\mathbf{Z} - \mathbf{R}||$  ;
7  R := Z ;
8  until  $a < \hat{a}$  ;
```

The experiments show that the algorithm has good convergence properties over hypertext structure of the Internet. For 322 million-link database the PageRank calculation converges to a reasonable tolerance in roughly 52 iterations.

PageRank index has been successfully used for search engine output sorting. Two such experimental search engines have been constructed at Stanford University, one of them is available for public use [W3], [2]. There seems to be quite high correlation between high PageRank index, and general page importance judged by human users. This is especially spectacular for very general queries, for which a lot of relevant pages exist in the Internet. PageRank succeeds in such situations by separating highly respected sites from junk pages that only happened to contain words from the query.

Other possible applications include web traffic estimation, citation count approximation and index personalization. It is assumed that this index could be used in other graph environments.

Two properties of the  $\hat{a}$  score function are of special importance. First, it is very simple, and quite inexpensive in calculation. Second, it corresponds well to human evaluation of pages, and is rather immune to commercial manipulation (if we want a page to have high index, we must convince creators of other highly ranked pages to add links to our page).

Finally, it has also very appealing interpretation. Appropriately normalized  $\hat{a}$  index can be interpreted as a probability measure of viewing a certain page by so called "random surfer", who randomly follows hyperlinks between web documents.

One of the most important drawbacks of this methodology is the necessity to have access to (ideally) entire Web structure to perform proper computation. This is possible only in large systems that maintain entire Web hyperlink databases such as general-purpose search engines and web crawlers.

---

<sup>1</sup> Selection of **S** seems to affect convergence speed, but little is yet known about selection techniques. One

## Web authorities and hubs - HITS algorithm

Web pages can serve various purposes, apart from just delivering information to the user. One of the most important types of pages found on the Internet are those, which facilitate navigation providing tables of contents, references and other types of outgoing hyperlinks. J. Kleinberg has noticed in 1997 [3] that applying only one ranking algorithm to a set of Web pages may produce misleading effects. For example, if we use ranking index based on number of back-links, such as  $\alpha$ , we can possibly identify pages delivering high quality information and therefore widely cited. However the pages containing important lists of references – being also of potentially high importance to the user - would not be ranked high. Kleinberg therefore has introduced two simultaneously computed indices, called *authority score* and *hub score*.

According to this methodology, in a set of documents generated by user query we encounter two types of pages. The first type pages, called *authority pages*, are highly relevant to initial query, and therefore should have considerable amount of back-links from other pages in retrieved set. The second type of pages in the retrieved set are those that point to many good authority pages. Such pages are called *hub pages*.

In other words a good *hub page* i.e. a page with high *hub score* is such a page, that points to many good authority pages, and similarly an *authority page* is such a page that is pointed to by many good hub pages. Ideally, given a query we should be able to isolate a group of relevant authority and hub pages "pulling together" and thus allowing throwing away any unrelated pages, those having large number of back-links (Fig. 3).

Let us describe the algorithm proposed by Kleinberg for computing hub and authority scores for a set of pages. By  $S$  we denote a root set of  $k$  pages that will be used as starting point in our analysis. Let  $L_p = (n_1, n_2, \dots, n_m)$  be a sequence of pages that are referenced by page  $p$  from above set  $S$ , and  $N(q)$  be the sequential number of page  $q$  in the sequence  $L_p$ . We define "one step neighbourhood"  $T'$  of set  $S$ , such that:

$$T' = \{t_i : t_i \in S \vee (\exists s_j \in S \ t_i \in L_{s_j} \wedge N(t_i) \leq d)\}$$

where  $d$  is an arbitrarily chosen number, so that size of  $T'$  is relatively small.

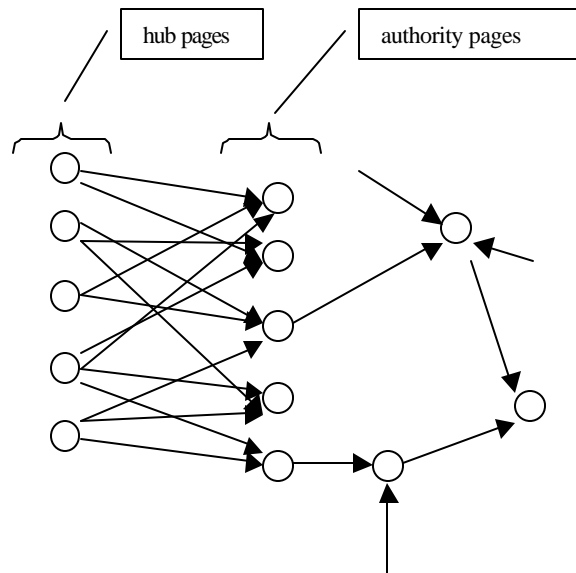
We define the *base set*  $T$  as a sum of sets  $S$  and  $T'$ :

$$T = S \cup T'$$

Let us introduce the sets  $V$  and  $E$ . The set  $V$  contains pages, and the set  $E$  contains links in form of ordered pairs  $(p, q)$  if the page  $p$  contains an *out-link* to page  $q$ . Let  $G(V, E)$  be a directed graph representing pages and hyperlinks from above set  $T$ . Then  $x_p$  be the authority score and  $y_p$  be the hub score for page  $p$  from  $V$ ;  $\mathbf{X}$  and  $\mathbf{Y}$  vectors over this set of web pages such as  $(x_1, x_2 \dots x_n)$ , and  $(y_1, y_2 \dots y_n)$  respectively, where  $n = |V|$ .

---

common practice is assigning  $\mathbf{S}$  the same values as for  $\mathbf{E}$



**Figure 3**

HITS(G) :

1  $\mathbf{X} := \mathbf{1}$ ;

2  $\mathbf{Y} := \mathbf{1}$ ;

3 Repeat

4 for  $i := 1$  to  $n$  do begin

5  $x_i := \sum_{q:(q,i) \in E} y_q$  ;

6  $y_i := \sum_{q:(i,q) \in E} x_q$  ;

7 end;

8 normalize  $\mathbf{X}$  and  $\mathbf{Y}$  so that  $\sum_{p \in V} x_p^2 = 1$  and  $\sum_{p \in V} y_p^2 = 1$  ;

9 until  $\mathbf{X}$  and  $\mathbf{Y}$  converge<sup>2</sup>;

Experiments performed in IBM Almaden Research Centre proved, that applied to real-world examples (i.e. sets of pages extracted from the Internet) this algorithm was able to isolate highly connected groups of pages sharing

<sup>2</sup> Kleinberg gives a mathematical proof of this algorithm convergence in [3]

common topic. Within these isolated groups, called *Web communities*, the authority score closely corresponds to other types or ranking indices, such as PageRank, and denotes page importance viewed by members of this particular community.

One of the advantages of this method is its ability to provide useful results even for a limited number of pages, for example obtained via a search engine query. To determine whether certain page has high *hub* or *authority score* we thus do not have to analyse entire web connectivity graph, which was necessary in PageRank method.

HITS method has also some other applications. One of the examples would be the problem of similar-page queries. Sometimes users have a need to find pages that are not defined by some kind of a query, but are semantically related or similar to certain page  $p$ . This could be achieved by applying the algorithm HITS with a slightly modified process of building the base set. Instead of creating the root set  $S$  from the search engine results, we can form it by using pages having hyperlinks pointing to the starting page  $p$ . The second part of the algorithm remains the same. For example for a page  $p$  being the home page of "*Honda Motor Company*", this approach produces such authority pages: "*Ford Motor Company*", "*Nissan*", "*Audi*", "*Toyota*", "*Dodge*". Detailed experiment results are described in [3]. Some other interesting proposals for application of HITS are structural analysis and topic distillation [4], [5].

## 2.2 Clustering pages using link topology

Exploiting information delivered by hyperlink network as described in the previous section can be sometimes risky. Ranking algorithms produce more or less universal indices, which averages other people opinion (represented by hyperlinks) about certain pages. Such an average may be sometimes controversial, and in certain cases - for example when we deal with a new concept or page that has not been noticed yet by other users - can produce bad results. Fortunately, we can use hyperlink information not only to approximate single page importance, but also to cluster similar or related pages into groups. This technique is called clustering or aggregation.

As ranking algorithms help us tame query result size problems, clustering methods allow us to decompose such problems into smaller ones. Separating large number of pages into semantically related clusters helps user to identify quickly which groups are relevant and important for him. He/she can then decide which groups should be removed from the search output. This can considerably reduce number of retrieved documents, and improve output readability.

One can note that clustering and ranking are not mutually exclusive. We can order pages within clusters using ranking algorithms, and even use ranking scores in a process of forming document aggregates.

## Classic graph-theory algorithms

One of the first attempts to use link information in clustering hypertext document repositories are experiments of Botafogo and Shneiderman, dating back to early nineties. In their paper [6] they have shown how classical metrics, such as graph compactness, could be used in identification of document types and their aggregation. They give a very simple, yet in some situations effective way to distinguish index and reference nodes in hypertext systems. *Index nodes* are such nodes in the hypertext graph, which purpose is the navigation

facilitation<sup>3</sup> while *reference nodes* deliver information. We denote by  $\bar{i}$  the mean number of out-links for all pages in a hypertext system, and by  $\sigma$  the standard deviation of number of out-links. Now:

*index node* is a node, which has more out-links than  $\bar{i} + \sigma$

*reference node* is a node, which has more back-links than  $\bar{i} + \sigma$

and  $\sigma$  is a threshold value, usually set to  $3\sigma$ , if numbers of links follow standard distribution.

The above measures are very coarse and work best in well-controlled hypertext systems. This means that they will probably fail in the Internet environment. However, thanks to its simplicity the method may come in hand in analysis of smaller WWW subsets, such as online books.

We will now describe an algorithm proposed in [6] for identification of groups of related pages.

Let  $n$  be the number of nodes in a hypertext graph  $H$ . Let  $C_{ij}$  be a distance between two nodes  $i$  and  $j$  from  $H$  such that:

$$C_{ii} = 0$$

$$C_{ij} = K, \text{ if there is no path between nodes } i \text{ and } j, K \text{ is arbitrarily set argument called } \textit{converted distance}$$

For the graph  $H$  we define compactness measure  $Cp(H)$  as follows:

$$Cp(H) = \frac{(n^2 - n)K - \sum_{i,j} C_{i,j}}{(n^2 - n)(K - 1)}$$

A graph  $H(V,E)$  is called *strongly connected component*, if for every  $i$  and  $j$  from  $V$  there are paths from  $j$  to  $i$  and from  $i$  to  $j$ . Now let graph  $H$  be a subgraph of a larger graph  $G$  representing a whole hypertext system. We call graph  $H$  a *semantic cluster* if  $Cp(H) > Cp(G)$ . According to Botafogo and Schneidman we assume that in well-built hypertext system strongly connected components of semantic clusters contain mainly closely related nodes, provided that links from *reference nodes* and links to *index nodes* have been deleted. So, the algorithm can be presented as follows.

decompose( $G$ ):

- 1 Repeat;
- 2 if  $G$  includes any *index* or *reference* nodes do begin
- 3 Remove out-links from index nodes
- 4 Remove back-links pointing to reference nodes
- 5 Treating  $G$  as undirected graph find biconnected components  $g_1 \dots g_n$ ;
- 6 if  $n=1$  return  $G$  as final component else

---

<sup>3</sup> Index nodes correspond roughly to Kleinberg's hub pages, while reference nodes can be treated as authority page prototypes

```

7   for i:=1 to n do decompose( $g_i$ );
8   end;
9   Until no index or reference nodes exist in  $G$ ;
10  decompose all final components into strongly connected components;

```

The algorithm has been tested on various hypertexts, all of them being well-controlled systems, handcrafted either by one author, or created under strong supervision of a central body. The results were quite encouraging, allowing the separation of groups having high compactness measure, which were mostly thematic groups of documents.

It seems that the above methodology in its pure form is rather not applicable to the Internet analysis. However, some of the ideas could be useful. For example incorporating such clustering mechanism in Web authoring tools could improve user friendliness and overall organization of large web sites (like corporate sites, or university web sites), which eventually could simplify document searching within them.

Yet another example of applying classic graph theory algorithms to hypertext environments is an algorithm of document similarity graph partitioning, proposed by Kazienko [7].

## Using HITS for clustering

The HITS algorithm described in section 2.1 can be viewed also as a clustering method. In fact it creates exactly *two* aggregates of pages from the input set - the pages with high authority and hub scores and the other ones - *junk* pages. Of course this may be not enough when the base set includes pages that can form more than one community of authorities and hubs. This may happen for example because we retrieve pages belonging to persons acting *pro* and *contra* certain idea, and therefore our set of pages consists of two *communities*.

Fortunately, only a slight modification of HITS algorithm<sup>4</sup> allows us to discover also such *non-principal communities*, and therefore partition the base set of pages into more than two groups.

Let  $\mathbf{A}$  be the adjacency matrix of a base set hyperlink graph, such that  $A_{u,v} = 1$  if there is a hyperlink from page  $u$  to page  $v$ , and  $A_{u,v} = 0$  otherwise. Let  $\mathbf{x}^*$  and  $\mathbf{y}^*$  be the vectors containing authority and hub scores for pages from the base set. Kleinberg shows in [3] that  $\mathbf{x}^*$  is the principal eigenvector of  $\mathbf{A}^T\mathbf{A}$ , and  $\mathbf{y}^*$  is the principal eigenvector of  $\mathbf{A}\mathbf{A}^T$ . The non-principal eigenvectors correspond to additional communities, and each pair of non-principal eigenvectors  $(\mathbf{x}_i^*, \mathbf{y}_i^*)$ , represents community of hubs and authorities that can be isolated from the base set. These communities have of course more sparse linkage structure than primary community, but experiments show that they deliver valuable information. For example using this technique to analyse Altavista search engine results for a query "jaguar jaguars" resulted in 3 communities related to "Atari Jaguar products", "NFL Jacksonville team" and "Jaguar automobiles". See [3] and [8].

In 1998 P.Raghavan [9] has implemented an approach similar to HITS as a part of Automatic Resource Compilation project at Stanford University. This project strives to build a system that could be able to automatically construct set of web resources for a given topic, which would be rated by human experts at least as

---

<sup>4</sup>Described in [3] and [8].

well as manually created indices such as Yahoo. The hub and authority algorithm has been augmented here by weighting links between pages according to the similarity of link text to the topic, for which an index is being built<sup>5</sup>. The exact formula for setting the link weight  $w(p,q)$  between pages  $p$  and  $q$  is as follows:

$$w(p,q) = 1 + n(t)$$

where  $n(t)$  denotes number of matches between terms in topic description and words from a page not farther away from `<a href=` tag than 50 characters. This exact distance has been selected by analysis of test web pages corpus.

According to [9], this method allows to quickly build reference lists of quality comparable to these created by human experts. A real advantage of ARC is the possibility of running it periodically in order to update once created indices, thus keeping them up to date - which is simply impossible for such repositories like Yahoo.

As we can see hyperlinks prove to be very powerful source of information, especially useful in such diversified environment as Internet. It is worth noting that the above methods perform actual ranking or clustering using almost no textual information, and in spite of that they are able to deliver spectacular results.

Some research has also been done in developing hybrid systems, utilizing both hypertext analysis and more traditional text analysis aided by thesaurus or category trees. Most interesting examples are Clever search system, developed in IBM Almaden Research Centre as an extension to HITS [5], and TAPER classifier<sup>6</sup> designed by Chakrabati, Dom and Indyk [10].

Interesting results have been also obtained by applying the spreading activation algorithm to graph structures of the Internet. We describe them in section 5.2 because the graphs involved are not only hyperlink graphs, but also text similarity and usage graphs.

## 2.3 Analysing hyperlink usage information

Classical hypertext systems contain often only two types of latent semantic information. These are hyperlink network and formatting information. However for some parts of the Internet - and especially for corporate Intranets - we can make use of one more type of information, namely, the traffic statistics. Thanks to Web Servers' logs we can analyse information about popularity of certain web pages among surfers. Such data can help searching and browsing processes in a variety of ways.

### Self modifying hypertext environments

Hyperlinks have been devised as a means of facilitating navigation among huge number of documents. User navigation patterns and habits are however variable, and hence hyperlink structure is determined only by Web service creators, it can sometimes be ineffective and lead to confusion. There is even special name for such hyperlink structure inefficiency effect, predating emergence of the World Wide Web: the *"lost in hyperspace phenomenon"*. Fortunately, latest advances in Web server and Web authoring technology make the Internet

---

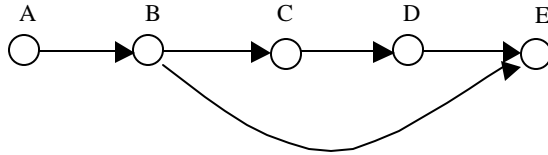
<sup>5</sup> And therefore contained within source code of page in proximity of `<a href="...">` and `</a>` tags.

<sup>6</sup> This system deals with a bit different problem though, which is categorizing hypertext documents into one of *predefined* classes

more and more flexible, so we can envisage such Internet services that would reconfigure its hyperlink structure in response to user navigation.

M. Merzbaher in [11] has been recently investigating the Dynamic Nearness algorithm in the context of such self-modifying systems. We will summarize this proposal below.

Consider for example the following hyperlink graph:



**Figure 4**

Now assume that the most common user browsing behaviour involves visiting node A, then B, then C, then B and finally  $E^7$ . It would be good to automatically create a direct link between pages B and E, and probably also between A and C if the above behaviour is common among large group of users. Note that this would not only ease the navigation, but also decrease the network traffic.

To determine the pages, between which a link should be generated, we use the Dynamic Nearness Algorithm, as presented in [11]:

addcounts(T):

- 1 Initialize all counts to 0;
- 2 for each transaction T do
- 3 for each distinct pair of pages U, V in T do
- 4 Increment the count between U and V;

Where *web transaction T* is a sequence of web pages that were accessed by the same user over short span of time. Such information is relatively easy to extract from Web server usage logs.

Next we use extracted *count* information to create a direct link between these two pages *U* and *V*, where count is higher than a given threshold value.

For more detailed overview of this method, together with experiments results see [11].

---

<sup>7</sup> Note that thanks to the caching mechanism built in into all contemporary Web browsers, the server log would not contain information about returning to page B.

## Spreading activation

In order to perform analysis of link topology, inter-page similarity and usage paths graphs for a World Wide Web subset Pirolli, Pitkov and Rao have proposed in their highly influential paper [12] a way of applying the Huberman and Hogg's *spreading activation* algorithm

The graph analysis is rather rarely used in the applications based on textual similarity. Note however, that classical document similarity matrices, like those discussed in Section 3, can be treated as graph incidence matrices - so we can actually talk about text similarity networks. Pitkov, Pirolli and Rao used three such matrices to represent various types of information related to WWW:

in *hyperlink topology matrix* an entry  $a_{i,j} = 1$  represents hyperlink presence between the pages  $i$  and  $j$

in *text similarity matrix* an entry  $a_{i,j} > 0$  represent the similarity strength between the pages  $i$  and  $j$

in *network usage matrix* an entry  $a_{i,j}$  is equal to the number of users that surfed from the page  $i$  to  $j$  in a given period

Above matrices have been interpreted as activation network incidence matrices. In such a matrix  $\mathbf{R}$  the value  $r_{i,j}$  represents how much *activation* flows from node  $i$  to node  $j$ . Let  $\mathbf{C}$  be a vector representing nodes pumping activation into network, such that  $C_i$  represents activation pumped by node  $i$ . The network behaviour can be then modelled iteratively, by computing the vector  $\mathbf{A}(t)$ , such that  $A_i(t)$  represents activation at node  $i$  at step  $t$ :

$$\mathbf{A}(t) = \mathbf{C} + \mathbf{M}\mathbf{A}(t-1)$$

where  $\mathbf{M}$  is a matrix determining flow and decay among nodes:

$$\mathbf{M} = (1 - \bar{g})\mathbf{1} + \bar{a} \mathbf{R}$$

where  $\bar{a} < 1$  is a parameter determining relaxation of node activity to zero level (when there is no activation flowing from neighbouring nodes) and  $\bar{g}$  is a parameter determining the amount of activation transmitted to neighbouring nodes.

In the experiments by Pitkov, Pirolli and Rao these parameters have been selected manually so that  $\frac{\bar{a}}{\bar{g}} \ll 1$ .

The discussed algorithm can be used in various ways. For example it can be used to determine which pages are most frequently visited from a given page. We build vector  $\mathbf{C}$ , such that it contains only one source of activation corresponding to the selected page, and set matrix  $\mathbf{R}$  to be the network usage matrix as defined above. We can then compute activations for limited number of iterations (for example 10, as proposed by Pitkov group) and then return most active pages as a web aggregate related to starting page. Other applications of this algorithm have been presented in [12].

### 3. Text mining related methods

Traditional methods for information retrieval in full text databases - and World Wide Web is a good example of such database - need a huge amount of external knowledge. Such knowledge repositories supporting document retrieval process can have many forms - like thesauri, stemming algorithms or dictionaries for translation - yet they share one common property. All have to be prepared by human experts, and therefore must be dramatically reduced in scope in comparison with amount of knowledge necessary to actually understand the meaning of documents. It seems to be one of the most important reasons, for which classical text retrieval methods fail in Internet environment. The World Wide Web contains documents on extremely heterogeneous topics, requiring extremely broad knowledge in the analysis process.

Fortunately, a sufficiently large document corpus contains lots of latent information, that can be extracted - or, if one prefers, mined - from the documents themselves. Such information, encompassing mainly cooccurrence and formatting data, proves to be very valuable in improving search process.

#### *3.1 Using cooccurrence analysis for page clustering*

Dumais and Landauer in 1990 have implemented Latent Semantic Analysis (LSA) in the form of statistical analysis of text [13]. They tested how "statistical" LSA improves search accuracy, by retrieving documents based on semantic content of words, rather than just simple word matching. The strength of LSA lies in its ability to represent textual atoms - such as documents, paragraphs, sentences and words - in a multidimensional semantic space. This representation embraces the correlation between words, documents and other lexical units, and therefore can be either used directly for searching (by constructing vector in a semantic space corresponding to user query) or for document clustering. Apart from being a very effective document analysis method, LSA seems to be a good model of some of the aspects of human learning process.

In order to process a set of documents<sup>8</sup> LSA generates a cooccurrence matrix, recording number of occurrences of each word in every analysed document<sup>9</sup>.

The next step of the analysis involves applying of the Singular Value Decomposition algorithm to decompose original matrix into product of smaller matrices. Finally, a reduced matrix containing generalized knowledge about lexical units' correlations is reconstructed.

We now define LSA more formally. Let  $\mathbf{A}$  be an  $m$  by  $n$  word-context occurrence matrix, such that:  $m \gg n$ ,

$\text{rank}(\mathbf{A}) = r$  and

$$a_{i,j} = L(i, j) \cdot G(i)$$

Where  $L(i,j)$  is a weighted measure of word  $i$  presence in the context of  $j$ , while  $G(i)$  is a global weighting function for words.

---

<sup>8</sup>These could be for example web pages retrieved by a search engine

<sup>9</sup>It is possible too to construct such a matrix using smaller lexical units, such as sentences.

Matrix  $\mathbf{A}$  is now decomposed by Singular Value Decomposition<sup>10</sup> into product of three matrices:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

where  $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}_n$ ;  $\mathbf{D} = \text{diag}(\mathbf{s}_1 \dots \mathbf{s}_n)$ ;  $\mathbf{s}_i > 0$  for  $1 \leq i \leq r$ ;  $\mathbf{s}_j = 0$  for  $j \geq r + 1$

Note that matrix  $\mathbf{U}$  has rows corresponding to the rows of matrix  $\mathbf{A}$ , and therefore to documents' words, and columns corresponding to newly derived variables (called factors or dimensions), such that each column is linearly independent of others. The  $\mathbf{V}^T$  matrix has columns corresponding to the original columns (in this example - contexts, or documents), and rows composed of derived singular vectors. Matrix  $\mathbf{D}$  is diagonal and contains values representing relation between  $\mathbf{U}$  and  $\mathbf{V}^T$  factors.

One of the properties of SVD is that, if some of the factors are removed - i.e. one or more values from  $\mathbf{D}$  are changed to 0, leaving other  $\mathbf{k}$  elements unchanged - the resulting matrix  $\mathbf{A}_k$  is a least-squares best approximation of the original. Therefore by reducing the number of factors we can capture the underlying structure of the word and document cocurrence, while at the same time we remove noise and variability in word usage. The effectiveness of this "purification" is largely affected by the parameter  $\mathbf{k}$ . It seems that the best effects are obtained when reduced number of dimensions corresponds to the semantic space dimensionality of original information source. See [14] for experiments results and [15] for more detailed examples of the SVD computation.

Michael Berry gives in [15] a good example of this phenomenon. Consider the words *car*, *automobile*, *driver* and *elephant*. If a classical retrieval system were not equipped with a thesaurus, it would not be able to match the query *car* to documents containing words *automobile* or *driver*. However, as words *car* and *automobile* have similar meaning, they are likely to occur in the similar contexts in the analysed documents<sup>11</sup>. Therefore, they will have similar representation in semantic space (they will have similar row representation in reconstructed matrix  $\mathbf{A}_k$ ), which can be exploited by LSA based search system. Similarly the word *driver* will be recognized as a bit less related and word *elephant*, which is unlikely to appear in the same context, would be regarded as entirely different. Of course such analysis works best, if the number of documents analysed is sufficiently large, for discussion on this see [1].

The similarity measures between documents discovered by LSA<sup>12</sup> can be used in classical clustering techniques to create clusters of similar documents, for example to reduce number of documents retrieved by a search engine. In the same time, the similarities between words could be used for automatic query broadening or as a base for computer aided query construction system.

Let us note that also the user query can be represented in the semantic space. The query, like a document, is a set of words, so we can add it to the input matrix  $\mathbf{A}$  before the SVD decomposition, and then check correlation between document vectors, and the query vector. We will not discuss here other areas of application of LSA, yet it is worth mentioning that it has been successfully used for such tasks like matching abstracts to full papers,

<sup>10</sup> Exact description of SVD can be found in [15].

<sup>11</sup> This is second-order cocurrence, as opposed to first-order cocurrence, where words appear closely (coocur) one to each other directly

<sup>12</sup> In most cases the similarity measure is computed as the cosine between generated vectors

matching reviewers to the submitted articles based on the analysis of reviewers own papers, text coherence, and quality assessment and cross-language retrieval.

As noticed by Landauer and Dumais [13] LSA also seems to be a good foundation for human memory and learning models. Especially it explains the problem dating back to Plato, which is that humans apparently have more knowledge then it has been presented to them. If the human learning process contains elements similar to LSA, we may be able to extract latent knowledge from the documents that we read and profit from the synergy effect introduced by the cocurence analysis. This knowledge may be not directly available if we examine documents separately. A very interesting and detailed analysis of this theory can be found in [13].

### 3.2 Recognition of formatting structures

Existing, publicly available search engines either do not process formatting information encoded in the HTML code, or even treat it as an obstacle and try to remove it, purifying page contents i.e. trying to produce more or less uniform text representation ready for further analysis. There have been some research conducted in creating such retrieval systems, that would intelligently perform such purification by reorganizing formatting structures in Web Page (such as lists or tables) and then converting data encoded in them into internal database representation that can be queried afterwards [16]. Such systems rely on manually created structure recognition engines called "wrappers" that have to be tailored for every new type of Web page.

We think however, that formatting information contains a lot of clues about the type of a page, and even about its contents. William Cohen gives in [17] the following example, which illustrates the potential strength of using formatting information in document retrieval:

Exploding porpoises, over four score and seven, well before configuration.

- *Department of Computer and Information Sciences*, University of New Jersey. Citrus flavorings: green, marine, clean and under lien.
- *Computer Engineering Center*, Lough Polytechnical Institute This, that page extensionally left to rights of manatees.
- *Electrical Engineering and Computer Science Dept*, Bismark State College. Tertiary; where cola substitutes are frequently underutilized.

This page under construction. (Last update: 9/23/98.)

Even though the actual text contained in this example is semantic and grammatical nonsense, we have *feeling* that this is part of a web page, listing university departments, together with their faculties, and probably also with their descriptions.

Above example is a bit crude - we still need to understand the words *university* and *department* to be able to deduct above conclusion. However it is quite easy to devise other examples where almost no such external information would be necessary. An alphabetical list of entries, containing email addresses (easily recognizable thanks to the @ character) and a lot of numerical characters (possibly phone numbers) is likely to be a contacts

list. Similar list, but separated into sub-lists for every alphabet letter, with a lot of *ftp* hyperlinks can be classified with high level of confidence as a software download list, etc.

One of the interesting efforts to create a system that would exploit such information is described in the aforementioned paper by W. Cohen [17]. A logic programming system WHIRL is used to encode the properties of two types of formatting structures - *simple list*, containing only bulleted entries, and *hot list*, that contains data on two attributes. This encoding is then used to automatically create *wrappers* that are able to adapt to new formatting scheme and extract meaningful data from such lists. Extracting knowledge encoded in the document format seems to be largely unexplored, yet very promising area. For example, [18] and [19] present some of new research in this field. The most obvious approaches of utilizing this knowledge, which are worth considering, are: clustering of similarly formatted documents and document decomposition into sub-components. One can also think of creating a document filtering system that would recognize "*junk documents*", that are not appropriately formatted, and therefore are probably "*off-topic*". Such approach should work especially well in extracting scientific or technical papers out of heterogeneous data sources, such as World Wide Web.

## 4. Query construction systems

Practically in all modern search engines the search process can be easily divided into two components. The first one is query building, where the computer system tries to get to know *what exactly* user wants to find. The second one involves actually finding this information. The query construction process is a very important step, as it determines what the retrieval engine would do, and if its quality is poor, the entire information retrieval system delivers unsatisfactory results. This is especially important in slow Web environment where the user can not repeatedly fine-tune the query because the system response time is usually very poor. Therefore the query construction system (or query language) should be:

- intuitive, so that the users do not have to waste time reading help pages
- powerful, allowing one to represent desired concepts
- uniform, that is consistent between various search engines

It is quite clear that the most appropriate query building system would be the natural language input. Unfortunately the current state of the art in the area of natural language processing and artificial intelligence renders any attempts to create reliable search system with natural language interface impossible. Nevertheless it is worth mentioning one of the systems that seems to achieve promising effects. This system - "*Ask Jeeves*" search engine [W4] - uses a huge knowledge base of parametric queries created by human experts. The user interface allows natural language input in a form of "asking a question to the system". The knowledge base is then browsed for most similar queries, which are returned, and can be executed. Each query has a well-defined set of associated terms (if a classical search engine such as Infoseek must be used to answer it) or just a list of URLs containing related information selected by human expert. If a user's query is more or less "direct" hit and matches the query from knowledge base the results can be spectacular. Alas, currently only a fraction of all queries can lead to such satisfactory effects (the system is constantly updated, mainly by processing user provided queries that were not found in the knowledge base).

Although we still cannot process *NL* queries effectively, we can implement alternative methods such as structured query languages or automatic factorisation systems.

## 4.1 Structured query languages

There are lots of data mining methods for typical relational tables. Classification techniques, association rules, neural nets and statistics are widely used in many commercial and freeware systems. Possibly, web mining can profit from this abundance of methodologies and algorithms. The main problem here is how to map Web structure and contents into a relational-like database. The first step towards utilising such approach seems to consist in building SQL-like languages for querying the World Wide Web.

Most of the work on such languages are based on the metaphor of Web as a huge, distributed database, albeit Web is not a database, and querying it is significantly different from querying a conventional database. For instance, Web has neither concurrency control nor transaction mechanisms. This may cause problems with computability of queries – it is easy to imagine a chain of links pointing to a document that is grown by a webmaster faster than we can follow them.

Despite these problems there are several attempts to adapt SQL and Datalog for Web purposes. According to Mendelzon and Milo [20] such adaptation should begin with formalization of the WWW structure. The formalization must take into consideration documents and links, as well as their contents. All the operations and imposed constraints have to be formally defined, too. In this way we build a formal model of Web as a database. At that point, according to this formal model one can set up syntax of a new querying language.

### W3QL

Konopnicki and Shmueli in [21] have proposed a high level SQL-like language named W3QL and a system W3QS that executes queries in this language. The idea behind this project was to go beyond limitations of traditional index servers (such as *Altavista*). The WWW is treated as a platform, a vehicle, in some sense similar to an operating system. On that platform there is virtual and distributed database that may be handled and managed by W3QL.

The formalization of the information space for W3QL includes many definitions of components for the whole WWW. It starts with important assumption that Web structure, contents and programs (CGI, ISAPI and applets) are static and programs are deterministic and time independent. Generally it is assumed that Web does not change during the execution of a query.

Another assumption is the view of Web as a directed graph. WWW documents are nodes and links are edges. The entire graph topology is unknown, but may be partially deduced by navigation activity. After few basic definitions of a WWW graph node, edges and parsing function – mapping each node into a set of edges, Konopnicki defines two kinds of queries specific for the WWW:

- Structure specifying queries. Result of them is a set of subgraphs of the whole WWW graph. A subgraph must be similar to the query graph. In order to detect similarity, a definition of *similarity mapping* is introduced.

- Content queries. Such queries may not operate on hypertext in nodes only. Also contents of links may be queried. For that case the definition of *content condition* and *legal similarity* of graphs are introduced.

Of course these basic types may be combined to form more complex queries.

A group of definitions in [21] apply to the HTML forms in Web documents. Web forms are treated as constraints imposed on *legal similarities* of query graphs. In the implementation, known forms may be filled according to patterns included in the dictionaries of forms. Unknown forms may be recovered – they are learned by including them with filled entries into a dictionary. It is an interesting example of automating the Web navigation even over HTML forms.

The syntax of W3QI takes into account all above theoretical considerations. . The *select* clause specifies links and nodes to be selected from a graph recovered by the rest of the query. The query's *from* clause includes definition of the graph as a comma-separated list of distinct nodes' names and distinct edges' names. For instance *n1,l1,n2,l2,n3* specifies a path from node *n1* to node *n3* via edge *l1*, node *n2* and edge *l2*. There is also a special sort of paths in Web graphs, called *unbounded length paths*. Such paths are represented as self-edges in the graph. For example a graph specified as *n1, l1, (n2,l2), l3, n3* includes a construct *(n2,l2)* that means self-edge named *l2* from node *n2* to itself. The *where* clause may include many different kinds of conditions imposed on the queried graph and its contents. Domain conditions are differentiated into strong and weak ones. Strong domain conditions have the following form:

```
n in {http://www.ii.pw.edu.pl/~okoniews, http://www.ii.pw.edu.pl/~gawrysia}
```

that means that node *n* must be element of the set defined inside braces.

Weak domain conditions, such as:

```
n in {/ii\.pw/, /bolek/}
```

are specified by the set of PERL regular expressions that have to be satisfied by URL of the node *n*.

Content conditions are also evaluated using PERL language constructions. For example:

```
n1, n2 : PERLCOND `n1.content eq n2.content`
```

To evaluate such condition W3QS runs external program PERLCOND that evaluates expression ``n1.content eq n2.content``. The engine of W3QS can be extended by other programs for evaluating specific types of expressions.

In order to enable working with HTML forms, the *Fill statement* is introduced. For example line

```
Fill n1 as in MyDof with query="Pink Dots"
```

specifies that the form must be filled out as in dictionary file *MyDof* with the entry *query* filled with "Pink Dots".

If the form can not be found in the dictionary of forms, one can run another external program:

```
Run learnform n1 cs76:0 if n1 unknown in MyDof
```

The function `learnform` is executed at user's Xterminal and lets the user fill out the form for the first time, and saves it into the dictionary of forms.

Examples of W3QL queries:

```
1  Select
2  From n1, l1, (n2,l2), l3,n3
3  Where
4  n1 in {http://www.ii.pw.edu.pl}
5  l1 in {/ii.pw.edu.pl/}
6  l2 in {/ii.pw.edu.pl/}
7  n3: PERLCOND 'n3.format=~/.image/'
8  n3 in {/ii.pw.edu.pl/}
9  Using ISEARCHd -d 5 -l 1000
```

In this query we search for images accessible from pages that are mentioned at <http://www.ii.pw.edu.pl>. Line 2 defines a path in WWW structure starting with node n1 and ending with n3. There must be a node n2 and links l1 and l3 between them. (n2,l2) is an unbounded length path of pages accessible from n1. Line 3 maps node variable n1 to specific starting page. Lines 5 and 6, use a PERL expression in order to limit the links to those pages that contain string 'ii.pw.edu.pl'. Line 8 contains the same constraint for the node variable n3. The external program PERLCOND is used for expressing content condition in line 7. In order not to overload an HTTP server one can limit the search length to 1000 HTTP requests and the length of the explored path to 5. It is achieved by passing arguments to the remote search program ISEARCHd in line 9.

### *WebSQL*

WebSQL is another querying language that attempts to reach the meta-level of Web in a similar way to W3QL. It is described in the papers [20] and [22]. A formal model of the Web for this language was described by Mendelzon in [20]. The assumption of static Web is also essential in this model. WebSQL is a language for querying database that contains two relations: one for documents and one for links, what makes it easy to understand for plain SQL users. It also uses some notation for regular expressions to indicate paths in the WWW graph. Content search is executed using pages generated by the index servers. Program that parses and executes WebSQL queries has a list of valid index servers and can read links from them.

An easy example of WebSQL query is :

```
1  Select d.url, d.title
2  From Document d Such That
3  "www.ii.pw.edu.pl"=|=>|=>=> d
4  Where d.Title Contains "Pink Dots"
```

We search here in a relational table Document (line 2) for a page that is zero, one or two links away from www.ii.pw.edu.pl (line 3). There is also a content condition in line 4

#### *Applications*

SQL-like languages are possibly good medium for carrying on more excessive research on Web mining. For example scripts written in such languages may constitute a method for keeping up-to-date information from Web. It could be a good way to command network agents, known also as “Web robots” or “knowbots”, especially when filling out forms [21]. W3QL and WebSQL seem to be good tools for the ones who want to avoid the limitations of traditional index servers – for instance to retrieve not only contents but also a structural information.

And, last but not least, SQL-like languages may be good start for all the researchers that will try to apply methods of KDD (Knowledge Discovery in Databases) for such as abundant source of information as WWW is.

## *4.2 Thesauri based query factorisation*

Thesauri are often used in modern search systems, either as a tool for dealing with semantic ambiguities within text corpora, or during query construction, as a kind of knowledge base facilitating query term selection. While such systems are often valuable, they have one obvious drawback. To use a thesaurus, you must have it - and hence the manual thesauri construction is a laborious process, there are not many general-purpose thesauri on the market.

Hinrich Shutze and Jan Pedersen [23] have proposed in 1996 a method of constructing a thesaurus from a sufficiently broad text corpus together with possible applications of such *coocurrence thesaurus* for information retrieval. Their method is somewhat similar to Latent Semantic Analysis described in Section 3. The main difference is that while LSA tries to analyse word-document relations, the Shutze&Pedersen thesaurus must be created from word-word relations.

This means that standard LSA method of applying SVD to the input coocurrence matrix is impractical, as the computational complexity of SVD is proportional to  $n^2$ ,  $n$  being the number of words in this case. For the thesauri creation process we should consider almost all words from the corpus, except possibly small number of stop-words.

Shutze and Pedersen use therefore iterative clustering process, presented below<sup>13</sup>.

autothesaurus:

- 1 Create matrix A for 3000 medium<sup>14</sup> frequency words. The coocurrence is defined here using sliding window method: two words coocur if they fit within 40 character wide window *sliding* over document text.
- 2 Cluster A elements into 200 classes  $g_{a1}, g_{a2}, \dots, g_{a200}$ .

---

<sup>13</sup> Actual figures have been taken from an experiment over TIPSTER corpus

<sup>14</sup> from 2000 to 5000 coocurrences between pairs of words in entire TIPSTER corpus.

- 3 Create matrix **B**, so that it records the cocurrence within sliding window of 20000 most frequent words with words from groups  $\mathcal{G}_{a1}, \mathcal{G}_{a2}, \dots, \mathcal{G}_{a200}$ .
- 4 Cluster matrix **B** into 200 classes  $\mathcal{G}_{b1}, \mathcal{G}_{b2}, \dots, \mathcal{G}_{b200}$ .
- 5 Create matrix **C**<sup>15</sup> recording cocurrences between all words from corpus and classes  $\mathcal{G}_{b1}, \mathcal{G}_{b2}, \dots, \mathcal{G}_{b200}$ .
- 6 Decompose matrix **C** by SVD.
- 7 Build reduced dimensionality matrix **D**

The similarity between vectors from matrix **D** (calculated for example as cosine between vectors) gives us a measure of word semantic similarity. To create thesaurus we can for example take some of the most similar (up to a given threshold) words for each term.

For full description and experiment results see [23]. We will only present a sample of thesaurus created from TIPSTER that illustrates the usefulness of this method:

Term	Related terms
Accident	repair, faulty, personnel, accidents, mishaps, exhaust, injuries, sites
Treatment	drugs, syndrome, administer, study, procedure, aids
Tax	taxes, income, tax-payer, incentives, levies, corporate

Such a thesaurus can be invaluable as a help tool in an information retrieval system. Shutze and Pedersen present two possible applications in this area. The first one involves creating a query vector in a way similar to the LSA method, while the second one deals with word factorisation in a query.

*Word factor* is a group of semantically related words in a query. Consider for example a query:

"text analysis" "world wide web" "information retrieval" "TREC" "text mining" "internet"

It has been apparently created as a means of retrieving documents dealing with "*information retrieval in the Internet*". Of course if submitted to any web search engine it would result in a multitude of documents about Internet *per se*. Therefore, what we need here is a Boolean expression similar to this:

---

<sup>15</sup> Note that this matrix is relatively small (in this TIPSTER example this is 450000 x 200 elements) when compared with simple word-word cocurrence matrix (450000 x 450000 elements).

```
("text analysis" OR "information retrieval" OR "TREC" OR "text mining") AND  
("world wide web" OR "internet")
```

Using the cocurrence thesaurus we can cluster query terms into the above two groups, and then ensure that only documents ranking high on *both* groups are retrieved. Note that in existing search systems such result could be achieved only by appropriate manual query creation<sup>16</sup>, and this assumes that user has some knowledge of entered terms semantic similarity, which is not always true.

### 4.3 Post-processing of query results

No software tool has yet been able to surpass human brain's capability to perform categorisation and identify trends in large amounts of data. Therefore we do not have to (and actually should not) rely always on artificial intelligence of systems to perform searching. Sometimes, assuming that a flexible and powerful user interface is given at our disposal, we can group, analyse and asses large amounts of documents by ourselves. Below we present two approaches to creating interfaces, which give users "broader view" over complex data structures either returned by in response to a query, or directly emerging during browsing process.

#### Hyperbolic trees

Browsing through large hypertext structures - such as World Wide Web - involves following large number of hyperlinks. Efficiency of this process is therefore determined by speed of changing context between pages, which in the Internet environment is directly dependent of download speed. This creates a barrier for users, especially when they browse large and complex web sites. To get the information they need, they must follow long hyperlink paths, sometimes without any clue that they are proceeding in the right direction in the hyperspace.

The concept of *site map* has been devised as a partial solution to this problem. Indeed, when we have a general overview of entire web site structure, we can in most cases quickly identify parts most interesting for us. Consider for example the process of finding the collection of *foreign sculpture* on the Louvre Web site (without search engine assistance, just starting from Louvre main page). To get to the desired page, for which a hyperlink is not directly available from main page, we must check pages such as "*collections*", "*temporary exhibitions*" and perhaps "*general information*", while on the general site map we can quite easily spot the page named '*foreign sculpture*'. This simple approach saves time and decreases network traffic, but unfortunately is static. Site maps must be manually created, and to be useful they must be also maintained, what can be laborious and costly for large sites.

Recently several companies, such as *Inxight*, started developing systems that would assist users in browsing by analysing the neighbourhood of currently viewed page, and producing dynamic equivalent of a site map. Inxight's approach, described in detail on their web page [W6] is unique also because created maps vary the graphical elements weight (such as size or hyperlink's line thickness) depending on user selection, therefore

---

<sup>16</sup> Several systems in which user have direct access to thesaurus during query construction are however available. See [24], [25] and [W5].

"zooming " on that part of web structure that is currently within focus of browsing process. Below we present aforementioned part of Louvre web site, with sculpture department clearly visible, after zooming onto collections page:

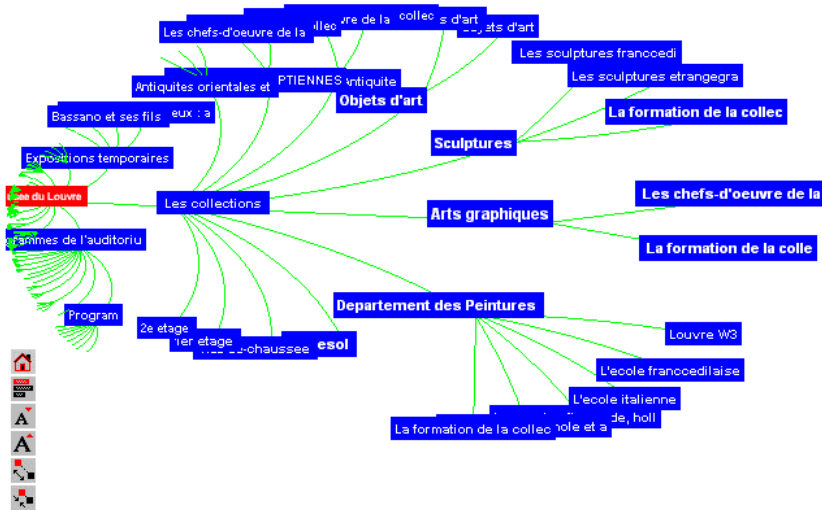


Figure 5

Note that such approach, while being very cheap to implement, may dramatically enhance browsing effectiveness, and therefore information retrieval results. Several other companies are also experimenting with similar visualization systems, sometimes combining them with text analysis tools. See *Semio* web page [W7] for another example of such system.

### Self-organizing document maps

Other approaches to giving users more general overviews of large document collections use the *semantic map* concept. Such map depicts the relationships between documents retrieved by a search engine in a form of a topographic map. The elevations on this map represent groups of strongly related documents (height corresponds to the relation strength and clustering properties), while low-level terrain represents unrelated documents creating *information noise*. Good examples of such approach are the *ThemeScope* semantic maps, which have been applied to Usenet newsgroups analysis. The effects can be observed on [W8].

Timo Honkela and Teuvo Kohonen have been running a similar project, called WEBSOM, at the Helsinki University of Technology. Their method relies on using neural networks concept - the self organizing maps - to represent relations between Usenet articles and to present them in a form of a browsable map. For detailed technical description, together with some comparisons of this method against LSA based clustering see [26] and [27], while the system demonstration is available at [W9].

## 5. Conclusion

In this paper we tried to present some new, non-conventional methods of document analysis that help one browse or retrieve documents from large collections available on Internet. All these methods have one feature in common - they do not use any external semantic information, and therefore do not try to perform traditional *natural language understanding* process. Instead they rely either on other than textual types of information (such as hyperlinks), or on its statistical analysis, and do not pretend to be a universal searching agent but rather enrich search process giving users tools for document analysis tasks such as grouping, categorisation and visualization.

Of course above methods are not perfect. First of all, purely link-based approach may be less and less efficient as Web creation strategies change. Recent trends in creating commercial sites involve incorporating as little *out-links* as possible, as they can lead to competitors web sites. In fact members of management and marketing communities now often use the term *click-away opportunity* instead of *hyperlink*<sup>17</sup>. This can have substantial effect on ranking indices such as Page Rank that judge page importance using hyperlink network density. However, systems similar to HITS should not be largely affected by this effect. Especially hybrid systems, such as ARC, should remain effective under these circumstances. We should mention here that HITS methodology is still being developed, recent advancements have been reported in [30] and [31].

Clustering methods, described in 2.2 above have a great potential for improving the capabilities of Internet search systems. Combined with clever visualization techniques, such as semantic maps, they can be very powerful tools in hands of an experienced user. In fact even more powerful user interfaces - such as SQL-like query languages, described in Section 4, can be very useful if users know how to exploit their potential. Entirely new area of *Internet usage education* has been emerged recently. It has not been covered by us here. Let us only mention that this is very important and largely overlook problem that deserves further research.

Innovative query languages and new query construction system seem to be very promising area of research. We do not think that a natural language query system will be possible in near future, yet any attempt to simplify search engine interfaces should have significant impact on their usefulness for casual users. The same applies for visualization systems that can help inexperienced people to interpret search engine output.

There is still much to be done in query language creation, as a uniform language has not yet been created. This is quite important for metasearchers i.e. systems that use many other search engines and process their results to produce one unified report. See Stanford University Digital Library Group [32] reports for an example of such universal query standard.

Text mining methods, such as LSA, are not likely to be implemented in public search systems, mainly because of their computational complexity. Their robustness is still not acceptable for such systems. However, we think that direct analysis of documents text will be the most important part of a research process in the future, when these problems are overcome. We must however remember that World Wide Web can be mined also for other types of multimedia information, for which even a textual description may not exist. This remains very interesting and almost unexplored research area - see [33] and [34] for more information about this subject.

There is no doubt that there is still a lot to discover in the area of *knowledge mining* in the Internet. Perhaps application of some new, constantly developed methods from related areas such as data mining, statistics or

---

<sup>17</sup> For other thoughts on future trends in Web searching see [28] and [29]

lexical analysis can bring about interesting results. Combining forces of these sciences may shed a new light on the Web mining research as well.

We think that intelligent information retrieval on the Internet and in large document repositories created by multinational corporations should be considered as one of the most important data management areas of research in the near future. There is still much to be done, but as we presented, even relatively unsophisticated methods can produce interesting results.

**Acknowledgements:** We would like to thank Prof. Henryk Rybiński and Prof. Mieczysław Muraszkiwicz for discussions and invaluable advice given during preparation of this paper.

## 6. Bibliography

- [1] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, "The Page Rank Citation Ranking: Bringing order to Web", Stanford University Digital Libraries papers, 1998
- [2] Sergey Brin, Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Proc. of 7<sup>th</sup> International Web Conference, 1998
- [3] Jon M. Kleinberg, "Authoritative Sources in Hyperlinked Environment", Proc. of Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998
- [4] D. Gibson, J. Kleinberg, P. Raghavan, "Structural Analysis of the World Wide Web", IBM Almaden Research Center, 1998
- [5] S. Chakrabati, Byron E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, "Experiments in Topic Distillation", IBM Almaden Research Center, 1998
- [6] Botafogo R.A., Schneiderman B., "Identifying aggregates in hypertext structures", Third ACM Conference on Hypertext, 1991
- [7] P. Kazienko, "Grupowanie stron WWW na podstawie odsy³aczy hipertekstowych", MISSI'98 conf. proceedings, Wroc³aw, 1998
- [8] D. Gibson, J. Kleinberg, P. Raghavan, "Inferring Web Communities from Link Topology", 9<sup>th</sup> ACM Conference on Hypertext and Hypermedia proceedings, 1998
- [9] P. Raghavan, "Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text", Proc. of 7<sup>th</sup> World Wide Web conference, 1998
- [10] S. Chakrabati, B. Dom, P. Indyk, "Enhanced hypertext categorisation using hyperlinks", ACM SIGMOD'98 proceedings, 1998
- [11] Matthew Merzbacher, "Discovering Semantic Proximity for Web Pages", ISMIS'99 conference proceedings, 1999
- [12] P. Pirolli, J.Pitkow, R.Rao, "Silk from a Sow's Ear: Extracting Usable Structures from Web", Proc. of the Conference on Human Factors in Computing Systems: Common Ground, pages 118-125, New York, 1996
- [13] T. Landauer, S.Dumais, "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge", Psychological Review, 1996

- [14] Laham D., Landauer T., Foltz P., "Introduction to Latent Semantic Analysis", *Discourse Processes* 25, pages 259-284, 1998
- [15] M. Berry, S. Dumais, T. Letshe, "Computational Methods for Intelligent Information Access", University of Tennessee, 1995
- [16] Z. Lacroix, A.Sahuguet, R. Chandrasekar "User oriented smart cache for Web: What You Seek is What You Get!", *ACM SIGMOD'98 proceedings*, 1998
- [17] W. Cohen, "Recognizing Structures in Web Pages using Similarity Queries", AT&T Research Labs, AAIL, 1999
- [18] D.W. Embley, Y.Jiang, Y.-K.Ng, "Record-Boundary Discovery in Web Documents", *Proc. of ACM SIGMOD International Conference on Management of Data*, 1999
- [19] H.Davulcu and others, "A Layered Architecture for Querying Dynamic Web Content", *Proc. of ACM SIGMOD International Conference on Management of Data*, 1999
- [20] A.O. Mendelzon, T. Milo, "Formal Models of Web Queries", *Proc. of ACM Symposium on Principles of Database Systems*, 1997.
- [21] D. Konopnicki, O.Shmueli, "Information Gathering in the World Wide Web: The W3QL Query Language and W3QS System", *ACM Transactions on Database Systems*, 1998
- [22] G. A. Mihaila, "WebSQL – An SQL-like Query Language for the World Wide Web", Master Thesis, Department of Computer Science, University of Toronto, 1996.
- [23] H. Shutze, J. Pedersen, "A Coocurrence based thesaurus and two applications to information retrieval", Xerox Palo Alto Research Center, 1996
- [24] M. Frelek, P. Gawrysiak, "Przeszukiwanie tekstowych baz danych w języku polskim", Master thesis, Warsaw University of Technology, 1998
- [25] M.Frelek, P.Gawrysiak, H.Rybiński, "A method of retrieval in flexion-based language text databases", *IIS'99 conf. proceedings*, 1999
- [26] T. Kohonen, "Self Organization of Very large Document Collections: State of the Art", *Proc. of 8<sup>th</sup> International Conference on Artificial Neural Networks*, 1998
- [27] T. Honkela, S. Kaski, K. Lagus, T. Kohonen "Exploration of Full-Text Databases with Self-Organizing Maps", Helsinki University of Technology, Neural Networks Research Centre, 1996
- [28] U. Manber. "Future Directions and Research Problems in the World Wide Web", *Proc. of ACM Symposium on Principles of Database Systems*, 1996.
- [29] P.Gawrysiak, "Using data mining methodology for text retrieval", *DIBS'99 conf. proceedings*, 1999
- [30] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, "Extracting large-scale knowledge bases from Web", *Proc. of 25<sup>th</sup> VLDB Conference*, 1999
- [31] S. Chakrabati, P. Raghavan and others, "Mining the Link Structure of the World Wide Web", IBM Almaden Research Center, 1999
- [32] Luis Gravano, Kevin Chang, Hector Garcia-Molina, Carl Lagoze, Andreas Paepcke, "Stanford Protocol Proposal for Internet Retrieval and Search", Stanford Digital Library working paper, 1997

[33] O. Zaiane, "Resource and knowledge discovery from the Internet and multimedia repositories", PhD Thesis Simon Fraser University, 1999

[34] O. Zaiane, J.Han, "WebML: Querying the World Wide Web for resources and knowledge", ACM CIKM'98 Workshop on Web Information and Data Management (WIDM'98), pages 9{12, Washington DC, November 1998.

[W1] [www.hotbot.com](http://www.hotbot.com)

[W2] [www.altavista.com](http://www.altavista.com)

[W3] [www.google.com](http://www.google.com)

[W4] [www.ask.com](http://www.ask.com)

[W5] [p5uni.i.pw.edu.pl/inferra](http://p5uni.i.pw.edu.pl/inferra)

[W6] [www.inxight.com](http://www.inxight.com)

[W7] [www.semio.com](http://www.semio.com)

[W8] [www.newsmaps.com](http://www.newsmaps.com)

[W9] [www.websom.hut.fi/websom](http://www.websom.hut.fi/websom)