NaviMoz: Mining Navigational Patterns in Portal Catalogs

Eleni Christodoulou, Theodore Dalamagas, and Timos Sellis

School of Electr. and Comp. Engineering
National Techn. University of Athens,
Athens, GR 15773
{hchristo,dalamag,timos}@dblab.ece.ntua.gr

Abstract. Portal Catalogs is a popular means of searching for information on the Web. They provide querying and browsing capabilities on data organized in a hierarchy, on a category/subcategory basis. This paper presents mining techniques on user navigational patterns in the hierarchies of portal catalogs. Specifically, we study and implement navigation retrieval methods and clustering tasks based on navigational patterns. The above mining tasks are quite useful for portal administrators, since they can be used to observe users' behavior, extract personal preferences and re-organize the structure of the portal to satisfy better user needs and navigational habits. These mining tasks have been implemented in the NaviMoz, a prototype system for mining navigational patterns in portal catalogs.

1 Introduction

Portal catalogs provide querying and browsing capabilities on data organized in a hierarchy on a category/subcategory basis. Users can navigate these hierarchies to identify data of their preference. Examples of such catalogs can be found in all popular search engines, e.g. in Google, Yahoo, OpenDirectory Project. Also, portal catalogs of a specific subject or domain (e.g. e-marketplaces for hardware, portals for cultural information) are provided by user communities on the Web. Such catalogs, known as vertical portals, is a valuable collection of resources for anyone who needs to search for information relevant to the interests of those communities. Portal catalogs maintain large volumes of information resources which is not possible for a single user to classify and exploit. For this reason, they are quite popular as a means for searching information on the Web.

A key point for maintaining portal catalogs is to observe users' behavior and extract personal preferences in order to re-organize the structure of the portal to satisfy better user needs. To support such tasks, current approaches [8, 10, 1] examine the pages that users visit by investigating the web logs of proxy servers. However, observing visited pages, without also paying attention to the categories in which these pages have been classified, cannot give an indication of the user *navigational habits*.

Navigational habits are related with how users search and browse in the paths of a hierarchy in a portal catalog. Paths in hierarchies provide a conceptual clustering of Web pages in groups sharing common properties. During a Web page search and browsing in a portal catalog, users may visit categories, go forward in subcategories if they look for pages with specific content or go backwards in more general categories if they look for Web pages with general content. We call *navigational patterns* the paths that users follow in the hierarchy of a portal catalog during a Web page search and browsing. Observing *navigational patterns* can give an indication of the concepts that users are interested in, and their navigational habits, and better support the maintenance of the portal catalog. For example, knowing that the navigational pattern `/gadgets/sound/mp3-players` is quite popular among users, the portal administrator may decide to put a link for the category `mp3-players` in the first level of the portal hierarchy to provide users with instant access to the list of mp3 players available. Also, discovering that many users go back and forth several times within a navigational pattern, might be an indication that the specific part of portal hierarchy is not well-designed and users cannot easily determine which category to follow. In this case, the administrator may decide to change that part, providing new categories or eliminating some of the old ones.

This paper develops mining techniques on user navigational patterns in the hierarchies of portal catalogs. Specifically, the work studies and implements navigation retrieval capabilities and clustering tasks based on common navigational patterns. The above mining tasks are quite useful for portal administrators for customizing the structure of portal catalog according to user and navigational habits. We next summarize the contributions of our work:

1. We introduce navigational patterns for hierarchies of portal catalogs.
2. We suggest a metric to capture the structure and content similarity between two navigational patterns.
3. We study and implement mining tasks for user navigational patterns in the hierarchies of portal catalogs. We develop navigation retrieval methods, and clustering tasks based on navigational patterns. In navigation retrieval methods, given navigational patterns as input, we determine users that follow navigational patterns which have common characteristics compared to those input patterns. In clustering tasks, we determine user groups that share common navigational habits.
4. The above tasks are implemented in the NaviMoz, a prototype system for mining navigational patterns in portal catalogs.

1.1 Related Work

Bioinformatics is one of the research fields where mining patterns (similar to navigational patterns that we study here) is a popular research issue. Biological entities, such as proteins and molecules, consist of sequences of elements, in the same way that the navigational patterns consist of sequences of categories of the portal hierarchy [7, 9, 5]. However, in this field, user requirements are different

and, thus, mining tasks needed are different, too (e.g. one can ask for the co-factors of a reaction) [7].

Also, there are many approaches that exploit Web logs of proxy servers to observe users behavior in Internet sites [4, 8, 10, 11, 17]. They are based on the words related to the hyper-links that users click and on the keywords related to the target Web pages. Some of those approaches [2, 10] assist the creation of user communities, that is groups of users with similar interests. In [17], a new algorithm is presented, which supports sequence discovery from multidimensional data, allowing the detection of sequences across monitored attributes, such as URLs and http referees. As noted in the introduction, observing visited pages, without also paying attention to the categories in which these pages have been classified, cannot give an indication of the user navigational habits, as studied in this paper. Thus, we believe that our work is complementary to the approaches that exploit Web logs. For a detailed discussion on web mining techniques for web personalization see [1].

The shape definition language (SLD) presented in [18] is also related to our work. The SDL language is used to retrieve objects based on shapes contained in the histories associated with these objects. The notion of shapes is close to the notion of navigational patterns.

The rest of the paper is organized as follows. Section 2 introduces navigational patterns in the hierarchies of portal catalog, and defines a similarity metric for navigational patterns. Section 3 presents the mining tasks developed. Section 4 describes our prototype system, and finally, Section 5 concludes the paper.

2 Hierarchies and Navigational Patterns in Portal Catalogs

Portal catalogs classify information resources in a semantically meaningful way. Their purpose is to develop and maintain specific communities of interests on the Web. Portal catalogs maintain large volumes of information resources, organized in thematic categories. The overall structure of a portal catalog is actually a hierarchy on a category/subcategory basis.

We can represent a hierarchy of a portal catalog as a graph structure $G = (V, E)$. V is the set of nodes representing categories included in the hierarchy, and E is the set of edges representing category/subcategory relationship. Since we represent a hierarchy of a portal catalog as a graph structure, a user can reach a category following different paths. For example, one reaches the category `History` through either `Science/Social Science/History` or through `Society/History`.

A user, during a Web page search and browsing in a portal catalog, may visit several categories. We call the sequence of categories in those visits `navigational patterns`. We note that such patterns may include multiple occurrences of categories. This might be the result of users going back and forth several times within a path in the graph of hierarchy.

A key issue for developing mining tasks for navigational patterns is to be able to estimate how similar two navigational patterns are. In the next subsection we present a similarity metric to estimate the similarity degree between two navigational patterns in terms of structure and content.

2.1 A Similarity Metric for Navigational Patterns

We design a similarity metric for navigational patterns in hierarchies of portal catalogs that takes into consideration both the structure of the pattern and the keywords used as labels for the categories.

1. To estimate the structure similarity between two navigational patterns, we consider the elements of navigational patterns as character sequences and we exploit the metric suggested in [12]. Such a metric is based on the minimum cost sequence of *edit operations* needed to change one string to become identical to another string. The set of edit operations include deletion of a character, insertion of a character and replacement of a character with another one. The calculation of the metric is based on a dynamic programming algorithm. The final result is the sum of the costs of the considered operations divided by the sum of the lengths of the navigational patterns. The result expresses a distance d. Thus, in order to have the similarity we should calculate $1 - d$.
2. To estimate the content similarity between two navigational patterns, we calculate the ratio of the number of occurrence of the common categories in both patterns to the total number of categories in both patterns.

The similarity metric is calculated as the average of structure and content similarity.

Let for example $A =$`/Health/Medicine/Fitness` and $B =$`/Health/Fitness/Running/Training/Coaching/Training` be two navigational patterns. Their structural distance is 0.55 (delete `Medicine` from A and insert `/Running/Training/Coaching/Training` : 5 operations, total length=9). Thus, their structural similarity is 0.45. The ratio of the number of occurrence of the common categories in both patterns to the total number of categories in both patterns is $4/9 = 0.44$. Thus, the similarity between these two navigational patterns is 0.445.

3 Mining Tasks

We study and implement mining tasks for user navigational patterns in the hierarchies of portal catalogs. Specifically, we develop navigation retrieval tasks and clustering tasks based on navigational patterns.

3.1 Navigation Retrieval Tasks

In all navigational retrieval tasks, a navigational pattern is given as input. Based on this input pattern, we can determine users that follow navigational patterns

with common characteristics compared to the input pattern. All tasks can be performed for a certain time period provided by the user. Specifically, we have developed the following navigation retrieval tasks:

- *Retrieval of navigations which are supersets of the input pattern.* This task identifies users whose navigational patterns contain all the categories from the input pattern (but these are not the only ones in the user pattern), keeping their ordering. For example, given that `/Arts/Radio` is an input pattern, the following navigational patterns will be part of the answer:
  - `/Arts/Radio/Personalities/Henrie_Phil/Personalities/Radio/Personalities/Programs/Voice_Actors`
  - `/Arts/Radio/Guides/Directories/Directories`
- *Retrieval of navigations which are subsets of the input pattern.* This task identifies users whose navigational patterns contain only categories from the input pattern (but these are not the only ones in the user pattern), keeping their ordering. For example, given that `/Arts/Radio/Guides` is an input pattern, the following navigational patterns will be part of the answer:
  - `/Arts/Radio`
  - `/Arts/Guides`
- *Retrieval of navigations which are identical to the input pattern.* This task identifies users whose navigational patterns contain all the categories from the input pattern (and these are the only ones in the user pattern), keeping their ordering.
- *Retrieval of the navigations which are similar to the input pattern.* This task identifies users whose navigational patterns are similar to the input pattern, given a similarity threshold. The threshold is provided by the user. A navigational pattern is retrieved as part of the answer if the similarity metric (introduced in the previous section) between itself and the input pattern gives a value that exceeds the threshold provided. We note that the suggested similarity metric takes into consideration both the structure of the pattern and the keywords used as labels for the categories. For example, given that `/Arts/Music` is an input pattern, and the threshold is 0.70, the pattern `/Arts/Radio/Music` will be retrieved (similarity=0.81), but the pattern `/Arts/Radio` will not (similarity=0.62).

3.2 Clustering Tasks

In clustering tasks, we can determine user groups that share common navigational habits. All tasks can be performed for a certain time period provided by the user. Specifically, we have developed the following tasks:

- *Grouping users that follow similar navigational patterns.* To support this task we have implemented two clustering algorithms: the K-means and the single link hierarchical clustering algorithm [13, 14]. For both clustering algorithms we exploit the similarity metric suggested in the previous section.

– *Retrieval of the most popular navigations.* This task identifies the navigational patterns which have been followed by the majority of users. We consider such patterns as popular patterns.
– *Retrieval of the most undecided users.* This task identifies users whose navigational habits indicate that they are undecided during their search and browse in the portal hierarchy. Also, it ranks the users according to how much undecided they are. We suppose that when a user goes back and forth during searching and browsing, he/she is an undecided user. An example of a navigational pattern which shows that the respective user is undecided is the following: `/Arts/Music/Pop/Music/Pop/Concerts/Pop/Music/Rock/Concerts`. A user whose navigational pattern is `/Arts/Music/Rock/Music/Rock/Concerts` is less undecided than the former, due to lower number of *back and forth* ($B\&F$) movements.

Next, we give an overview of clustering techniques and we discuss in detail the single link implementation. The key point of our implementation is that the estimation of the clustering level for single link is performed exploiting the C-index method [16].

Clustering Algorithms Clustering methods are usually divided into two broad categories. *Non-hierarchical methods* group a data set into a number of clusters. *Hierarchical methods* produce nested sets of data (hierarchies), in which pairs of elements or clusters are successively linked until every element in the data set becomes connected. Non-hierarchical methods have low computational requirements, ($O(kn)$, if for example n documents need to be grouped into k clusters), but certain parameters like the number of formed clusters must be known a priori. Hierarchical methods are computationally expensive, with time requirements of $O(n^2)$, if n documents need to be clustered. However, hierarchical methods have been used extensively as a means of increasing the effectiveness and efficiency of retrieval [20–22]. For a wide ranging overview of clustering methods one can refer to [13, 14]. *Single link*, *complete link* and *group average link* are known as hierarchical clustering methods. All these methods are based on a similar idea:

1. Each element of the data set to be clustered is considered to be a single cluster.
2. The clusters with the minimum distance (i.e. maximum similarity) are merged and the distance between the remaining clusters and the new, merged one is recalculated.
3. While there are more than one clusters, go again to step 2.

In single link (complete link), the distance between two non-single clusters is defined as the minimum (maximum) of the distances between all pairs of elements so that one element is in the first cluster and the other element is in the second cluster. In group average link, the distance between two non-single clusters is defined as the mean of the distances between all pairs of elements

so that one element is in the one cluster and the other element is in the other cluster.

We implemented a single link clustering algorithm using Prim's algorithm [23] for computing the *minimum spanning tree (MST)* of a graph. Given a graph G with a set of weighted edges E and a set of vertices V, a MST is an acyclic subset $T \subseteq E$ that links all the vertices and whose total weight $W(T)$ (the sum of the weights for the edges in T) is minimized. It has been shown [24] that a MST contains all the information needed in order to perform single link clustering.

Given n navigational patterns, we form a fully connected graph G with n vertices $\in V$ and $n(n-1)/2$ weighted edges $\in E$. The weight of an edge corresponds to the similarity distance between the vertices (trees) that this edge connects. The single link clusters for a *clustering level* l_1 can be identified by deleting all the edges with weight $w \geq l_1$ from the MST of G. The connected components of the remaining graph are the single link clusters. Figure 1(a) shows a graph with 7 nodes that correspond to 7 navigational patterns, and 10 edges. The weight of an edge is the similarity distance between the involved navigational patterns. For example the similarity distance between patterns 1 and 2 is 0.2. The missing edges, that is the extra edges that make the graph fully connected, are those that have weight 1. Figure 1(b) shows the minimum spanning tree of (a). Figure 1(c) presents the graph remaining after deleting all edges with weight ≥ 0.4. There are 2 connected components that include nodes $(1, 2, 3, 6)$ and nodes $(7, 5)$, respectively. This indicates the presence of 2 clusters: cluster 1 with $(1, 2, 3, 6)$ as members and cluster 2 with $(7, 5)$ as members. Nodes which are not connected to other nodes will be considered as single-node clusters.

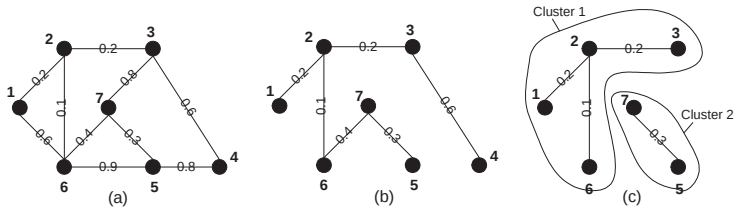

Fig. 1. Minimum Spanning Tree (MST) detection and single link clustering at level 0.6.

A stopping rule is necessary to determine the most appropriate clustering level for the single link hierarchies. Milligan et al. present 30 such rules [25]. Among these rules, $C-$index [16] exhibits excellent performance (found in the top 3 stopping rules). We next present the way we adopt the $C-$index in a hierarchical clustering procedure for navigational patterns.

***C*−index for Hierarchical Clustering** $C-$index is a vector of pairs $((i_1, n_1), (i_2, n_2), \ldots, (i_p, n_p))$, where $i_1, i_2, \ldots, i_p$ are the values of the index and n_1,

$n_2, \ldots, n_p$ the number of clusters in each clustering arrangement produced by varying the clustering level of a hierarchical clustering procedure in p different steps. Let l_1 be the first selected clustering level, which produces an arrangement of N_1 clusters (that is $n_1 = N_1$): C_1 with c_1 elements, C_2 with c_2 elements, ..., C_{N_1} with c_{N_1} elements. We can calculate i_1 in order to have the first pair (i_1, n_1) of C-index vector:

$$i_1 = (d_w - min(d_w))/(max(d_w) - min(d_w))$$

where:

1. $d_w = Sum(d_{w_1}) + Sum(d_{w_2}) + \ldots + Sum(d_{w_{N_1}})$, with $Sum(d_{w_i})$ to be the sum of pairwise similaritites of all members of cluster $C_i, 1 \leq i \leq n_1$,
2. $max(d_w)$: the sum of the n_d highest pairwise similaritites in the whole set of data (that is, sort distances, highest first, and take the Top-n_d sum),
3. $min(d_w)$: the sum of the n_d lowest pairwise similaritites in the whole set of data (that is, sort distances, highest first, and take the Bottom-n_d sum),

with $n_d = c_1 * (c_1 - 1)/2 + c_2 * (c_2 - 1)/2 + \ldots + c_{N_1} * (c_{N_1} - 1)/2$ (that is the number of all within cluster pairwise similaritites). Similarly we calculate all values of $C-$index for all different p clustering levels, getting the vector $((i_1, n_1), (i_2, n_2), \ldots, (i_p, n_p))$. We point out that:

- Although all pairwise similaritites are needed to compute the C-Index, this doesn't require any additional computation because these similaritites need to be computed anyway for the hierarchical clustering procedure itself.
- Since multiple successive clustering levels can generate the same number of clusters, we compute the $C-$Index not for each level but for each number of clusters generated by different levels.
- The number of clusters with the lowest $C-$Index is chosen as the correct clustering, as [25] suggests.

We next present the algorithm exploited to retrieve the most undecided users.

Undecided Users To support this task, we have implemented an algorithm which counts how many $B\&F$ movements a user has made. The term $B\&F$ refers to a pair of categories in the hierarchy of a portal catalog. A pair of categories (A, B) is $B\&F$ if:

- Both A and B belong to the same navigational pattern P.
- $label(A) = label(B)$: the categories are the same.
- There is an odd number of categories (different than A and B) between A and B in pattern P.
- $label(after(A)) = label(before(B))$ (after(A) gives the category which is after A in the examined navigational pattern, while before(A) gives the category which is before A in the examined navigational pattern).

According to the above, a $B\&F$ pair appears in the primitive navigational pattern `A/B/A`. We call such $B\&F$ pairs as *basic* $B\&F$s, since the patterns in which can appear are of minimum length. The algorithm first detects the basic $B\&F$s, and then the others.

Input: navigational pattern
Output: number of the $B\&F$ pairs in the pattern.
Algorithm:
array BF: For every category of the input pattern, it contains its $B\&F$ partner (i.e., the category with which it constitutes a $B\&F$).
Vector P: Contains all category pairs (A, B) which are candidates for $B\&F$s, and the number of categories that exist between A and B.
Vectors valueVector: One vector for each category, having the place where the category occurs in the navigational pattern.
int counter: Contains the number of $B\&F$.

/*Find basic B&F */

```
for all categories
   if the size of the current category valueVector >1
      Check whether there are two successive occurrences in
      places i and j of the category in the pattern, such that
      no other category exists between places i and j.
      If there are not such successive occurrences then
         if one category exists between places i and j.
            B&F exists for category in i and category in j
            Add B&F to array BF
         else
            if N categories exist between places i and j,
            with N odd
               candidate B&F exists for category in i and
                    category in j
               add this category pair to Vector P

Sort Vector P according to the number of categories that exist
between categories which constitute a candidate B&F.
```

/*Find the rest B&F */

```
for every pair (A,B) of the sorted Vector P
   Check if the categories after(A) and before(B) form a pair in
   array BF, and if so, the pair (A,B) is B&F.
      Update array BF
   else
      Find the category in array BF that constitutes a B&F pair
            together with after(A)
      If that category is before(B)
```

```
      Add the B&F pair to array BF
   else
      Find the category in array BF that constitutes a B&F pair
         together with BF[after(A)]
```

/*Computation of the total number of B&F */

```
For every non-zero element of array BF, increase counter by 1.
Return counter.
```

An example is given in case the input to the algorithm is the navigational pattern: `r/s/m/d/m/o/m/s/f/s/f/w/f/w`. The algorithm first detects the basic $B\&F$s, which are the category pairs $(2, 4)$[1] (i.e., m and m), $(4, 6)$, $(11, 13)$, $(7, 9)$, $(8, 10)$ and $(10, 12)$ (i.e., f and f), and updates the array BF . Vector P keeps the category pair $(1, 7)$ as a candidate $B\&F$. The algorithm examines the category pair $(2, 6)$ and finds that there exist $B\&F$ pairs $(1, 4)$ and $(4, 6)$. Thus, the pair $(1, 7)$ is $B\&F$.

3.3 User Identification Tasks

These tasks provide useful information regarding user identification. Specifically, the system provides (a) user navigation retrieval for a given time period, and (b) user session retrieval for a given time period.

4 System Description

The above tasks are implemented in the NaviMoz, a prototype system for mining navigational patterns in portal catalogs. NaviMoz system consists of three basic modules:

1. User Manager Module. This module provides user login operations and maintains the user access to the portal catalog. The users' navigations are stored in the database and they are further explored by the system's manager.
2. Mining Module. This module provides all the mining tasks for navigational patterns described in this paper.
3. Storage Module. This module is responsible for maintaining an RDBMS used for storing users information and their navigational patterns.
4. Presentation Module. This module supports the graphical interface of NaviMoz.

The architecture of NaviMoz is presented in Figure 2, while a screen dump showing a clustering task running is illustrated in Figure 3. Clustering has identified a cluster involving three users (Christodoulou Eleni, Kalimerh Maria and Kanellakopoulos Haralampos) whose navigational habits are similar.

[1] numbers denote places in the navigational pattern

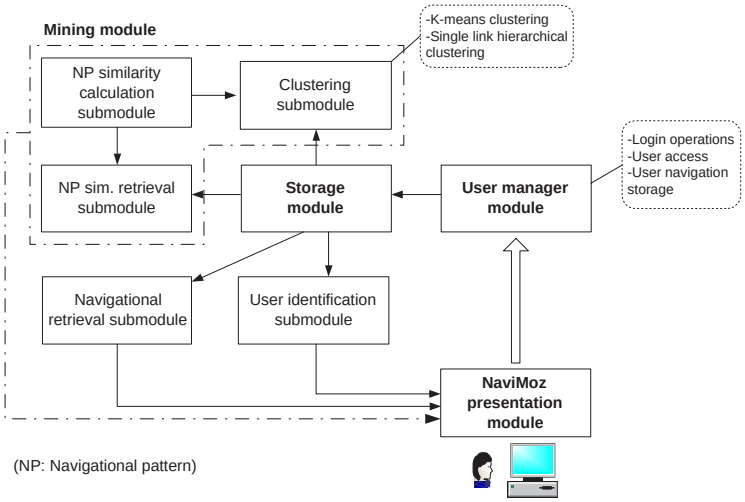

Fig. 2. Architecture of NaviMoz.

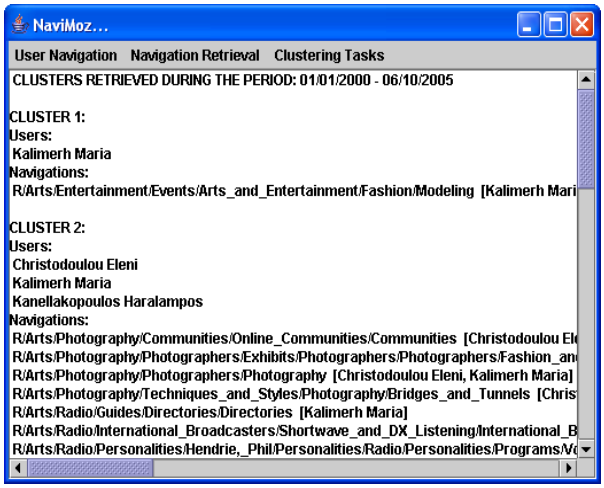

Fig. 3. Clustering task in NaviMoz.

5 Conclusions

Observing user behavior and extracting personal preferences is crucial for maintaining portal catalogs. Certain navigational habits can indicate the need to re-organize the structure of the portal to satisfy better user needs. Observing

visited pages in a portal catalog, without also paying attention to the categories in which these pages have been classified, cannot give an indication of the user navigational habits.

This paper suggests a set of mining tasks on user navigational patterns in the hierarchies of portal catalogs. Those mining tasks can help portal administrators for customizing the structure of portal catalog according to user and navigational habits. In our work, we introduced navigational patterns for hierarchies of portal catalogs, and we designed a metric to capture their structure and content similarity. Based on this metric, we implemented several mining tasks, like navigation retrieval methods and clustering methods, for user navigational patterns in the hierarchies of portal catalogs.

The navigational patterns can be considered as simplified Xpath queries [19]. We will extend our work to employ complex navigational patterns (e.g., branching path expressions), expressed as Xpath pattern queries. Our future plan for NaviMoz is to become a full-fledged generic portal management system that will provide editing operations fully supported by the mining tasks described in this paper.

References

1. M. Eirinaki, M. Vazirgiannis, Web mining for web personalization. ACM Transactions on Internet Technology (TOIT), 3(1), 2003.
2. C. R. Anderson, E. Horvitz. A Dynamic Personalized Start Page. In Proceedings of the 11th WWW Conference 2002.
3. A. Ajijth, V. Ramos. Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming. Congress on Evolutionary Computation, IEEE Press ISBN 078-0378-04-0 pp 1384-1391, Canberra, Australia, Dec 2003.
4. X. Fang, O. R. Liu Sheng. Designing a Better Web Portal for Digital Government: A Web-Mining Based Approach, http://diggov.org/library/library/dgo2005/demosb/fang_designing.pdf
5. Y. Kaneta, M. Md. Munna Ahaduzzaman, T. Ohkawa. A Method of Extracting Sentences Related to Protein Interaction from Literature using a Structure Database. In Proceedings of the 2nd European Workshop on Data Mining and Text Mining for Bioinformatics (ECML/PKDD'04), Italy, September 2004.
6. T. Kamdar. A. Joshi. On Creating Adaptive Web Sites using Web Log Mining, TR-CS-00-05. Department of Computer Science and Electrical Engineering University of Maryland, Baltimore Country, (2000).
7. L. Krishnamurthy, J. Nadeau, G. Ozsoyoglu, M. Ozsoyoglu, G. Schaeffer, M. Tasan, W. Xu . Pathways Database System: An integrated set of tools for biological pathways. Bioinformatics 19(8) 2003.
8. B. Mobasher, H. Dai, T. Luo, Y. Sung, J. Zhu.: Integrating Web Usage and Content Mining for More Effective Personalization. In Proceedings of the International Conference on E-Commerce and Web Technologies, Greenwich, UK, 165-176, 2000.
9. R. G. Pensa, C. Leschi, J. Besson and J. Boulicaut. Assessment of discretization techniques for relevant pattern discovery from gene expression data. In Proceedings of the 2nd Workshop on Data Mining in Bioinformatics, Seattle, USA, August 2004.

10. D. Pierrakos, G. Paliouras, C. Papatheodorou, V. Karakaletsis and M. Dikaiakos. Web community directories: A new approach to web personalization. In Lecture Notes in Artificial Intelligence (LNAI), 3209, Springer-Verlag, 2004.
11. F. Toolan, N. Kusmerick. Mining web logs for personalized site maps. In Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'02), 2002.
12. R. A. Wagner, M. J. Fischer, The String to String Correction Problem, Journal of the Association for the Computer Machinery, Vol 21, No.1, pp.168-173, January 1974.
13. E. Rasmussen, Clustering algorithms, in: W. Frakes, R. Baeza-Yates (Eds.), Information Retrieval: Data Structures and Algorithms, Prentice Hall, 1992.
14. M. Halkidi, Y. Batistakis, M. Vazirgiannis, Clustering algorithms and validity measures. In Proceedings of the SSDBM Conference, Virginia, USA, 2001.
15. T. Dalamagas, T. Cheng, K.J. Winkel, T. Sellis, A Methodology for Clustering XML Documents by Structure, Information Systems, Elsevier, 2004.
16. L. J. Hubert, J. R. Levin, A general statistical framework for accessing categorical clustering in free recall, Psychological Bulletin 83 (1976) 1072–1082.
17. Matthias Baumgarten, Alex G. Buchner, Sarabjot S. Anand, Maurice D. Mulvenna, John G. Hughes: User-Driven Navigation Pattern Discovery from Internet Data. 74-91
18. Agrawal, R., Psaila, G., Wimmers, E.L., Zat, M.: Querying shapes of histories. In: Proceedings of 21st International Conference on Very Large Data Bases, Morgan Kaufmann (1995) 502–514
19. XML path language (XPath: www.w3.org/TR/xpath)
20. N. Jardine, C. J. van Rijsbergen, The use of hierarchical clustering in information retrieval, Information storage and retrieval 7 (1971) 217–240.
21. E. Voorhees, The effectiveness and efficiency of agglomerative hierarchic clustering in document retrieval, Ph.D. thesis, Cornell University, Ithaca, New York (Oct. 1985).
22. M. Hearst, J. O. Pedersen, Reexamining the cluster hypothesis: Scatter/gather on retrieval results, in: Proceedings of the ACM SIGIR Conference, Zurich, Switzerland, 1996, pp. 76–84.
23. T. Cormen, C. Leiserson, R. Rivest, Introduction to algorithms, MIT Press, 1990.
24. J. C. Gower, G. J. S. Ross, Minimum spanning trees and single linkage cluster analysis, Applied Statistics 18 (1969) 54–64.
25. G. W. Milligan, M. C. Cooper, An examination of procedures for determining the number of clusters in a data set, Psychometrika 50 (1985) 159–179.