Association for Information Systems
AIS Electronic Library (AISeL)

PACIS 2004 Proceedings

Pacific Asia Conference on Information Systems (PACIS)

December 2004

Hyperlink Structure-based Recommender System

Ying Zhou
Fudan University

Youwei Wang
Fudan University

Yueyuan Zhang
Alcatel Shanghai Bell Corp. Ltd

Weihui Dai
Fudan University

Hyperlink Structure-based Recommender System

Ying ZHOU
Management School
Fudan University
032025059@fudan.edu.cn

Youwei WANG
Management School
Fudan University
ywwang@fudan.edu.cn

Yueyuan ZHANG
Alcatel Shanghai Bell
Corp. Ltd
Yueyuan.Zhang@alcatel-sbell.com

Weihui DAI
Management School
Fudan University
whdai@fudan.edu.cn

Abstract

Recommender systems are applied widely in e-commerce and information access. They provide suggestions to users to prune unrelated information so that users are guided to those items that best meet their needs. A variety of techniques have been proposed for performing recommendation, including content-based, collaborative, knowledge-based and other techniques. However, hyperlink-structure based recommendation is neglected. This paper supplies the lack, and introduces a recommender system using the clustering and similarity of the hyperlink structure.

Keywords: Recommender system, Hyperlink structure, E-commerce

1. Introduction

Recommender systems were originally defined as ones in which "people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients" (Resnick et al. 1997). The term now refers to any system that provides personalized recommendations as output or guides the user in an individualized way to interesting objects in a large space of possible options. In the hyperlink environment, recommender systems are obviously very useful, because the amount of on-line information exceeds any individual's ability to handle it. A growing number of companies, including Amazon.com, CDNow.com and Levis.com, employ or provide recommender system solutions (Schafer et al. 1999).

Many researchers have worked on recommender systems in the last few years. Collaborative recommendation is probably the most familiar, most widely implemented and most mature of the technologies. Collaborative recommender systems aggregate ratings or recommendations of objects, recognize commonalities between users on the basis of their ratings, and generate new recommendations based on inter-user comparisons (Burke 2002). Grundy (Rich 1979) studied demographic recommender systems, aiming to categorize the user based on personal attributes and make recommendations based on demographic classes. Belkin and Croft (Belkin et al. 1992) addressed content-based recommendation, which is an outgrowth and continuation of information filtering research. A content-based recommender learns a profile of the user's interests based on the features present in objects the user has rated. Utility-based and knowledge-based recommenders do not attempt to build long-term generalizations about their users, but rather base their advice on an evaluation of the match between a user's need and the set of options available. Utility-based recommenders make suggestions based on a computation of the utility of each object for the user (Guttman et al. 1998). Knowledge-based recommendation attempts to suggest objects based on inferences about a user's needs and

preferences (Towle et al. 2000).

The approaches above mainly attempt to achieve better recommendation; however, they have neglected a simple way, using hyperlink structure to make recommendation. Since the hyperlink structure is built up basing on certain logic, such as the relationship between products, it is rational to make recommendation based on it. In this paper, we will explore to build up such a hyperlink structure-based recommender system. In order to reflect the individualization of users, we will introduce a mechanism using a memory pool to store access history of a specific user.

The remainder of this paper is organized as follows: section 2 describes the preliminary knowledge of the proposed recommendation technology, which includes the strongly connected components and the node similarity function; the proposed recommender system is addressed in section 3; section 4 gives out an example to show how the system functions; this paper concludes with summary and future directions.

2. Preliminaries

2.1 Index and Reference Nodes

As the name implies, *index nodes* are nodes that can be used as an index or guide to many other nodes. For example, a hypertext document that points to all the other documents is an index. Formally, an *index node* is a node whose outdegree is greater than the mean outdegree of all nodes, plus a threshold value.

Reference nodes usually are referred to by other nodes. For example, in a hypertext document about mobile phones, all CDMA mobile phones might link to an article about “CDMA”. Formally, a *reference node* is a node whose indegree is greater than the mean indegree of all nodes, plus a threshold value.

In a more precise way, we can define index and reference nodes based on a function of their out and in-degrees (Rodrigo et al. 1991).

Definitions

• Let μ be the mean of the outdegrees of the nodes in the hypertext and let μ' be the mean of the indegrees of the nodes in the hypertext. Note that $\mu = \mu'$ since every link that leaves a node has to reach another node. For this reason we will use μ for both means.
• Let σ be the standard deviation of the outdegrees of the nodes.
• Let σ’ be the standard deviation of the indegrees of the nodes.
• Let τ be a threshold value.
• An index node is a node whose outdegree is greater than $\mu + \tau$.
• A reference node is a node whose indegree is greater than $\mu' + \tau$.

We usually define τ as been equal to $3 * \sigma$ (σ’). The motivation for this choice is as follows: if the number of links follow a normal distribution, then the probability that a node has in or out-degree exceeding three standard deviations is less than one percent.

2.2 Biconnected Components and Strongly Connected Components

Definitions:

• Articulation point: If a node breaks the graph into more than one component when removed then it is called as articulation point As shown in fig. 1, node 2,6,7 and 9 are articulation points.

• Biconnected components: In a graph in a selected component, if you can go from one of the nodes to all other nodes by an alternative path then the component is called biconnected component. Articulation points separate the graph into biconnected components. In fig. 2, there are five biconnected components: {1,7,8}, {2,5,7,9}, {3}, {6} and {4}.

• Strongly connected components: Two nodes "a" and "b" are in the same strongly connected component if there is a path from node "a" to node "b" and a path from node "b" to node "a" (Rodrigo et al. 1991). As shown in fig.3, {1,7,8} is a strongly connected component.

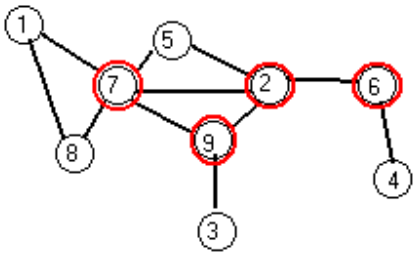

Fig.1 Articulation point

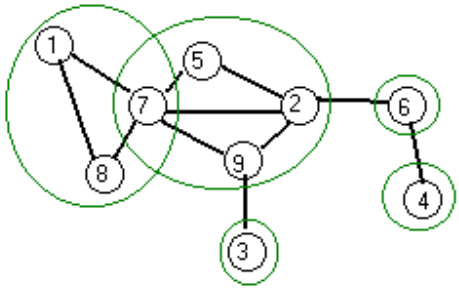

Fig.2 Biconnected components

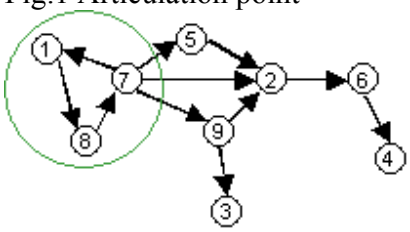

Fig.3 Strongly connected components

Biconnected components in a graph have the property that there are at least two paths between any two nodes in this component. Finding biconnected components is a quite simple task and can be implemented in O (V+E) (Robert 1983). Since at least two paths exist between two nodes in a biconnected component, it is likely that biconnected components will be semantic clusters of the hypertext. Strongly connected components further enhanced the clustering of nodes.

Using the notions described above, a clustering algorithm by interrelation of nodes can be developed:

Step 1) Find the index and reference nodes in the hypertext. If none exist and the algorithm has run at least once go to step 6.

Step 2) Remove outgoing edges from index nodes and incoming edges from reference nodes.

Step 3) Treating the graph G as undirected, find the biconnected components.

Step 4) Build the reduced graph G' from G.

Step 5) For each of the biconnected components go back to step 1.

Step 6) For each biconnected components left, decompose it into strongly connected components.

Some important features of this algorithm should be observed. First, the algorithm is recursive (step 5). This implies that every bicomponent found will be treated as an independent graph (with fewer nodes than the original graph) and consequently it will be possible to find new index and reference nodes. Finding those nodes and removing their links will allow for a more precise clustering of the hypertext with the intrinsic relationship between nodes assuming an important role.

The second important property of this algorithm lies in step 4. The reduced graph G' of G is a tree. This simplifies enormously the structure of the hypertext that goes from a complex graph to a very simple tree structure.

This clustering algorithm will be used to split the hyperlink structure into a set of strongly connected components. Then this set of strongly components and the following node similarity function will generate the final recommendation set.

2.3 Node similarity function

We use S_{ij} to describe the similarity between two nodes (Dhyani et al. 2002). It reflects three important notions about certain hyperlink structures that imply semantic relations: a path between two nodes, the number of ancestor nodes that refer to both nodes, and the number of descendant nodes that both nodes refer to (Ron Weiss et al. 1996).

For our discussion, we use the following definitions:
sp_{ij} : length of a shortest path between node i and node j
sp_{ij}^{k} : length of a shortest path between node i and node j not traversing node k

Shortest Paths

It is rational to hypothesize that the similarity between two nodes varies inversely with the length of the shortest path (Kilfoil et al. 2003) between the two nodes. Because the hypertext links are directional, we consider the shortest path both from node i to node j and from node j to node i. If there is no path between node i and node j, we do not add any weight to the similarity function for this component. We use S_{ij}^{sp} to describe the shortest path component of the node similarity function. It can be obtained through equation 1:

$$S_{ij}^{sp} = \frac{1}{2^{(sp_{ij})}} + \frac{1}{2^{(sp_{ji})}} \qquad (1)$$

The denominator ensures that as shortest paths increase in length, the similarity between the nodes decreases.

Common Ancestors

We hypothesize that the similarity between two nodes is proportional to the number of ancestors that the two nodes have in common. We use S_{ij}^{an} to describe the common ancestors component of the node similarity function. It can be obtained through equation 2:

$$S_{ij}^{an} = \sum_{\substack{k \in common \\ ancesters}} \frac{1}{2^{(sp_{ki}^{j} + sp_{kj}^{i})}} \qquad (2)$$

S_{ij}^{an} considers the length of the shortest path between a common ancestor and both node i and node j. As the shortest paths increase in length, the similarity decreases. Also, the more common ancestors, the higher the similarity.

The computation normalizes S_{ij}^{an} to lie between 0 and 1 before it is included in S_{ij}. The weight contribution from each ancestor k is divided by the number of ancestors in the same "level" as k. The level of k with respect to node i and node j is the minimum distance from either node i or node j.

A common ancestor a does not contribute to S_{ij}^{an} when the only path that reaches node j from a is through node i. If node i is an ancestor of node j, then all ancestors of node i are automatically ancestors of node j. A path between node i and node j is already considered in the similarity measurement with the S_{ij}^{sp} component. Therefore, S_{ij}^{an} does not include ancestors that are not proper common ancestors.

Common Descendants

The similarity between two nodes is also proportional to the number of descendants that the two nodes have in common. We use S_{ij}^{de} to describe the common descendants component of the node similarity function. It can be obtained through equation 3:

$$S_{ij}^{de} = \sum_{\substack{k \in common \\ descendants}} \frac{1}{2^{(sp_{ik}^{j} + sp_{jk}^{i})}} \qquad (3)$$

The computation normalizes S_{ij}^{de} to lie between 0 and 1 before it is included in S_{ij} in the same manner as the normalization for S_{ij}^{an}.

Complete Node Similarity

The complete node similarity function between two hyperlink nodes i and j, S_{ij}, is a linear combination of the above three components:

$$S_{ij} = W_s * S_{ij}^{sp} + W_a * S_{ij}^{an} + W_d * S_{ij}^{de} \qquad (4)$$

Here W_s, W_a and W_d represent the weights of the three components. They can be determined through expertise or through experimental data.

The node similarity function above is a bit complex. It's time complexity is O (n^3). But since it is used to generate the background data, it can be calculated offline. So it is relatively fast for examining a graph.

3. Proposed Recommender System

Specifically, recommender systems have (i) background data, the information that the system has before the recommendation process begins, (ii) input data, the information that user must communicate to the system in order to generate a recommendation, and (iii) an algorithm that combines background and input data to arrive at its suggestions (Burke 2002).

In our proposed recommender system, we will use the strongly connected components and the

node similarity function to generate background data. First the hyperlink structure is treated as a directed graph. Then we use the clustering algorithm introduced in section 2.2 to split the graph into a set of strongly connected components, that is also a set of semantic clusters of the hyperlink structure. Let scc_1, scc_2 …scc_m be the generated clusters and SCC is a set of them. Suppose a user accessed a node in scc_1, then we can recommend other nodes in this cluster to this user. In order to get a more precise recommendation, we have to calculate the similarity S_{ij} between any two nodes in the hyperlink structure. This can be done using the node similarity function mentioned in section 2.3. For each node, we can use S_{ij} to generate a set of high similarity nodes by setting a threshold t. For node i, if S_{ij} > t, then node j belongs to the high similarity set of node i. Here the threshold t can be obtained through historical data or from expertise. Let hss_1, hss_2 … hss_n be the generated high similarity sets and HSS is a set of them. Then we can use HSS to provide recommendation. For example, if a user accesses node i, then the nodes in hss_i can be recommended to this user. So far, the background data, SCC and HSS, is ready for further processing.

The recommendation process begins when a user enters the hyperlink structure by accessing any one of the nodes. Based on the first node the user accessed, we find two corresponding sets in SCC and HSS. All the nodes in these two sets are recommended to the user. For example, if a user first accessed node i, then we find sets hss_i and scc_j to which node i belongs. The union of these two sets is recommended to the user. When the user moves to another node, we can repeat the same procedures to make recommendation. It must be mentioned here that it is not necessary to use the union of hss_i and scc_j. If the number of links is very large, then the intersection of hss_i and scc_j will be a better choice.

When exploring nodes, different users have different accessing orders. This difference reflects users' preference of nodes to a certain extent. So we added a memory pool to our system, storing the last recommendation set. This memory pool can be viewed as a window. A proper length *l* is assigned to the memory pool to avoid the case that all nodes stay in it. When the user moves from the current node to another node, put all the nodes in the current recommendation set into the memory pool. If the number of nodes in current recommendation set is greater than *l*, select the nodes with the largest S_{ij} values into the memory pool. And the former nodes in the memory pool are pulled out.

The memory pool, together with the union of hss_i and scc_j make up the final recommendation set for the current node. Certainly, the existing links of the current node should be excluded.

Having described the details of our recommender system, we now summarize the system. We call it the HSB (for hyperlink structure-based) recommendation system. The following is a list of notations used in the system:

n	total number of nodes in the hyperlink structure
m	total number of strongly connected components

SCC	set of strongly connected components
scc_j	the j-th strongly connected component in SCC
S_{ij}	similarity between node i and node j
t	threshold for determine high similarity set
HSS	set of every node's high similarity set
hss_i	high similarity set of node i
mp	memory pool nodes set
l	length of the memory pool
C	existing links of current node
R	final recommendation set

Background data generation

(1) Use the algorithm proposed in section 2.2 to split all nodes into strongly connected components SCC = $\{scc_1, scc_2 \ldots scc_m\}$

(2) Compute the similarity S_{ij} between any two nodes in the hyperlink structure

(3) If $S_{ij} >$ t, put j into hss_i and i into hss_j to form the HSS = $\{ hss_1, hss_2 \ldots hss_n \}$

Recommendation process

(4) User accesses node i

(5) Find the strongly connected component scc_j to which the current node i belongs

(6) Calculate R = $scc_j \cup hss_i \cup$ mp – C

(7) If the number of nodes in R is less than l, set mp = R, or else select the nodes with the largest S_{ij} values into mp. Go back to step 4.

Fig.4. HSB recommender system

4. An Example

In this section, we will give out a simple example to illustrate how the HSB recommender system functions. The example hyperlink structure consists of 10 nodes and 18 links as in Fig.5:

Mean + Standard deviation 2.4 2.78

	In-degrees	Out-degrees
a	3	2
b	2	2
c	2	1
d	2	0
e	2	2
f	1	2
g	2	2
h	1	4
i	2	2
j	1	1
Mean(μ)	1.8	1.8
Standard deviation(σ)	0.6	0.98

Fig.6 Value characteristics of the example

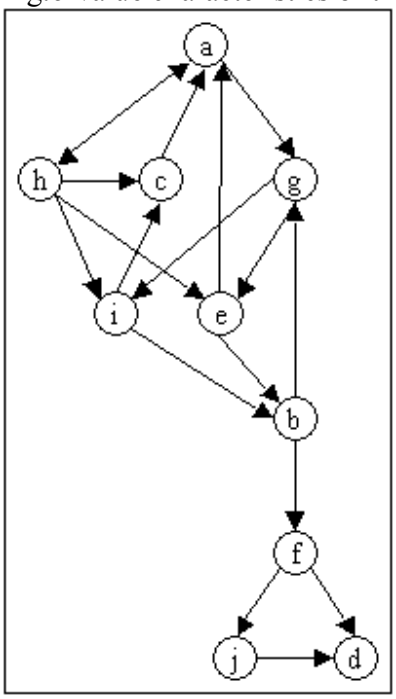

Fig.5 Example hyperlink structure

Here the threshold value for identifying articulation points is (μ + σ) instead of (μ+3*σ). This is because the number of nodes and links in this simple example is very small. When handling actual hyperlink structure, (μ+3*σ) will be better. Follow the algorithm presented in section 2.2, first we identified the index node h and reference node a. In Fig.8 we removed the outgoing edges of node h and incoming edges of node a. Using the algorithm of Robert Sedgewick, we separated the graph into four biconnected components in Fig.9. And in Fig.10, the graph was further split into six strongly connected components.

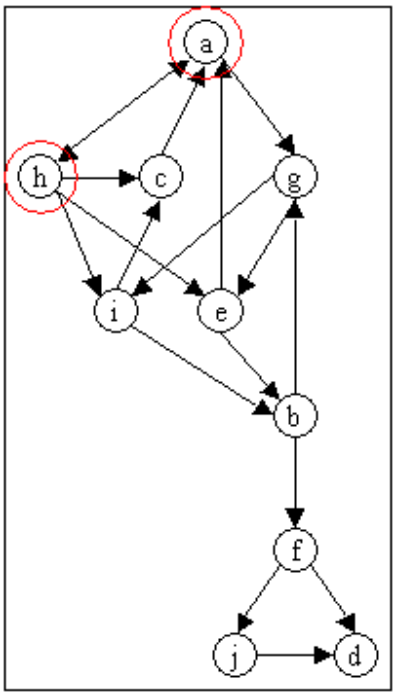

Fig.7 Find index and reference nodes

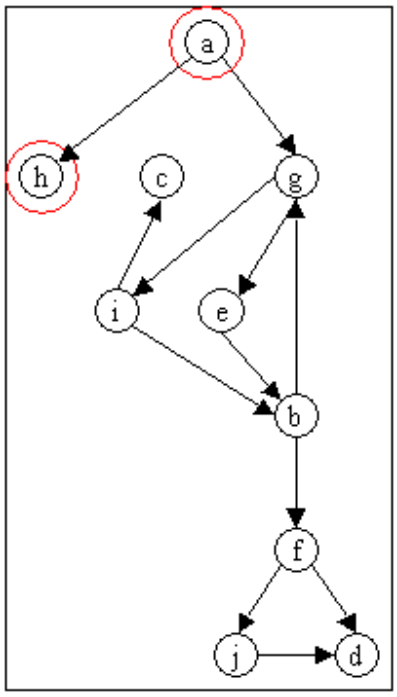

Fig.8 Remove edges

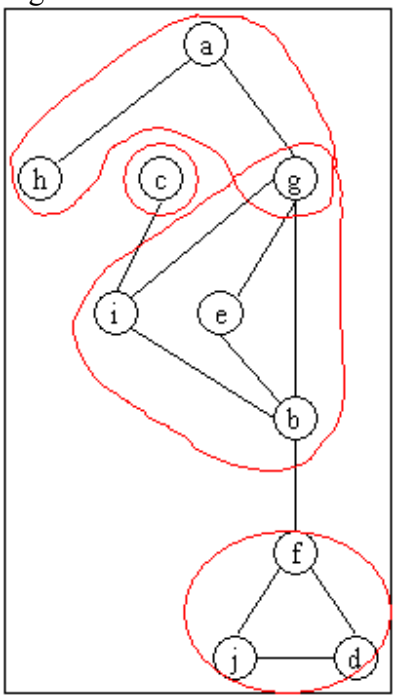

Fig.9 Find biconnected components

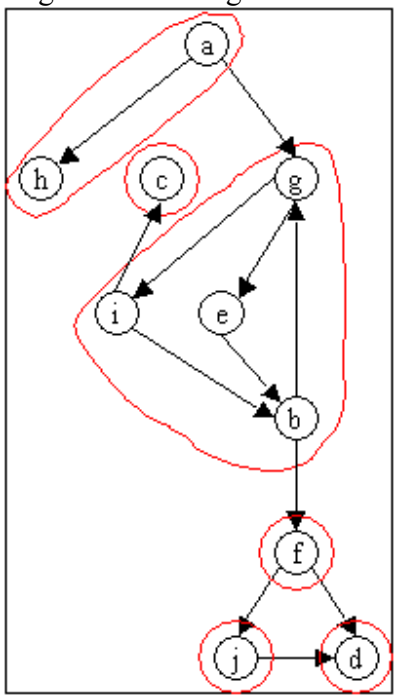

Fig.10 Find strongly connected components

Thus SCC = {(a, h); (c); (b, e, g, i); (f); (d); (j)}

To calculate the similarity between any two nodes, we use the following equation:

$S_{ij} = 1/3 * S_{ij}^{sp} + 1/3 * S_{ij}^{an} + 1/3 * S_{ij}^{de}$

For convenience's sake, the weights of the three components all equal 1/3. And the threshold

value for identifying high similarity nodes is set to 0.25. In actual environment, the weights and the threshold value must be obtained from historical data or from expertise. The final calculation results are summarized in Fig.11:

	a	b	c	d	e	f	g	h	i	j
a		0.406	0.380	0.078	0.425	0.135	0.452	0.507	0.445	0.073
b	0.406		0.297	0.083	0.357	0.167	0.365	0.370	0.411	0.125
c	0.380	0.297		0.060	0.405	0.099	0.403	0.445	0.384	0.063
d	0.078	0.083	0.060		0.076	0.167	0.095	0.051	0.092	0.286
e	0.425	0.357	0.405	0.076		0.184	0.469	0.513	0.505	0.113
f	0.135	0.167	0.099	0.167	0.184		0.180	0.102	0.184	0.250
g	0.452	0.365	0.403	0.095	0.469	0.180		0.539	0.458	0.105
h	0.507	0.370	0.445	0.051	0.513	0.102	0.539		0.521	0.061
i	0.445	0.411	0.384	0.092	0.505	0.184	0.458	0.521		0.113
j	0.073	0.125	0.063	0.286	0.113	0.250	0.105	0.061	0.113	

Fig.11 Similarity between any two nodes

Thus hss_a = (b, c, e , g, h, i)

hss_b = (a, c, e, g, h, i)

hss_c = (a, b, e, g, h, i)

hss_d = (j)

hss_e = (a, b, c, g, h, i)

hss_f = (j)

hss_g = (a, b, c, e, h, i)

hss_h = (a, b, c, e, g, i)

hss_i = (a, b, c, e, g, h)

hss_j = (d, f)

Then begins the recommendation process. The memory pool length is set to 4 here. Suppose the user explores the hyperlink structure in this order: a→c→i, then the HSB recommendation system functions as Fig. 12 shows:

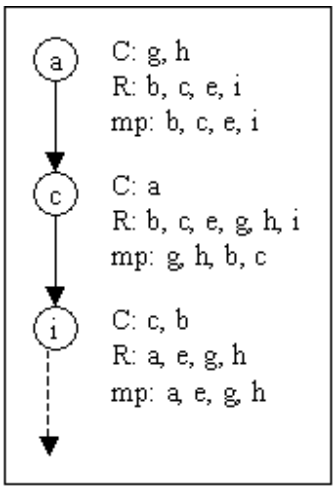

Fig.12 Recommendation process

5. Conclusion

In this paper, a system is addressed to make recommendations based on hyperlink structure. The following issues need to be solved for further research:

1) Experiments with actual websites should be carried out to validate the efficiency of the recommender system.
2) Parameters *l, t,* W_s, W_a and W_d need to be determined through analysis of historical data.
3) The impact analysis of number of nodes on the recommender system.

The long-term goal of the research is to develop a system that can automatically generates the final recommendation set. This may help the web site manager to grasp more insight about the hyperlink structure to improve the recommendation process. And based on this insight, optimization of the hyperlink structure could be carried out.

Acknowledgements

This research is partially sponsored by:
1) Libra arts research promotion program, Fudan University: Jinmiao Project (EXH1019325): fuzzy availability-based e-business website structure evaluation and optimization method
2) Youth scientific research funds, School of management, Fudan University (CHH1019070): e-business website updating technology research

References

Belkin, N. J. and Croft, W. B. “Information Filtering and Information Retrieval: Two Sides of the Same Coin?” *Communications of the ACM* 35(12), 1992, pp. 29-38.

Burke, R. “Hybrid recommender systems: Survey and experiments,” *User Modeling and User Adapted Interaction*, Vol.12 (4), 2002, pp. 331-370.

Dhyani, D., Ng, W. K., and Bhowmick, S. S. “A survey of web metrics,” *ACM Computing Surveys*, 34(4), 2002, pp. 469-503.

Guttman, R. H., Moukas, A. G. and Maes, P. “Agent-Mediated Electronic Commerce: A Survey,” *Knowledge Engineering Review*, 13 (2), 1998, pp. 147-159.

Kilfoil, M., Ghorbani, A., Xing, W., Lei, Z., Lu, J., Zhang, J., and Xu, X. “Toward an adaptive web: the state of the art and science,” *In Communication Networks and Services Research (CNSR) 2003 conference*, New Brunswick, CA, 2003, pp. 108-119,130

Resnick, P. and Varian, H. R. “Recommender Systems,” *Communications of the ACM*, 40 (3), 1997, pp. 56-58.

Rich, E. “User Modeling via Stereotypes,” *Cognitive Science* 3, 1979, pp. 329-354.

Robert Sedgewick “Algorithms,” *Addison Wesley Publications*, Chapter 30 - Connectivity, Biconnectivity, 1983, pp. 324-330.

Rodrigo A. Botafogo and Ben Shneiderman, “Identifying Aggregates in Hypertext Structures,” *Proceedings of the third annual ACM conference on Hypertext*, 1991, pp. 63-74.

Ron Weiss et al. “HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering,” *In Proceedings of the 7th ACM Conference on Hypertext*, 1996, pp. 180-193.

Schafer, J. B., Konstan, J. and Riedl, J. “Recommender Systems in e-Commerce,” In: EC ’99: *Proceedings of the First ACM Conference on Electronic Commerce*, Denver, CO, 1999, pp. 158-166.

Towle, B. and Quinn, C. “Knowledge Based Recommender Systems Using Explicit User Models,” In Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, *AAAI Technical Report* WS-00-04, Menlo Park, CA, AAAI Press, 2000, pp. 74-77.