Florida Institute of Technology
Scholarship Repository @ Florida Tech

Theses and Dissertations

5-2005

Learning Implicit User Interest Hierarchy for Web Personalization

Hyoung-rae Kim

Follow this and additional works at: https://repository.fit.edu/etd

 Part of the Computer Sciences Commons

Learning Implicit User Interest Hierarchy for Web Personalization

by
Hyoung-rae Kim

A dissertation submitted to
Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Computer Science

Melbourne, Florida
May 2005
TR-CS-2005_12

© Copyright 2005 Hyoung-rae Kim
All Rights Reserved

The author grants permission to make single copies _____

Learning Implicit User Interest Hierarchy for Web Personalization
a dissertation by
Hyoung-rae Kim

Approved as to style and content

Philip K. Chan, Ph.D.
Associate Professor, Computer Sciences
Dissertation Advisor

Debasis Mitra, Ph.D.
Associate Professor, Computer Sciences

Marius-Calin Silaghi, Ph.D.
Assistant Professor, Computer Sciences

Alan C. Leonard, Ph.D.
Professor, Biological Sciences

William D. Shoaff, Ph.D.
Associate Professor, Computer Sciences
Department Head

Abstract

Learning Implicit User Interest Hierarchy for Web Personalization

by

Hyoung-rae Kim

Dissertation Advisor: Philip K. Chan, Ph.D.

Most web search engines are designed to serve all users in a general way, without considering the interests of individual users. In contrast, personalized web search engines incorporate an individual user's interests when choosing relevant web pages to return. In order to provide a more robust context for personalization, a user interest hierarchy (UIH) is presented. The UIH extracts a continuum of general to specific user interests from web pages and generates a uniquely personalized order to search results.

This dissertation consists of five main parts. First, a divisive hierarchical clustering (DHC) algorithm is proposed to group words (topics) into a hierarchy where more general interests are represented by a larger set of words. Second, a variable-length phrase-finding (VPF) algorithm that finds meaningful phrases from a web page is introduced. Third, two new desirable properties that a correlation function should satisfy are proposed. These properties will help understand the general characteristics of a correlation function and help choose or devise correct correlation functions for an application domain. Fourth, methods are examined that (re)rank the results from a search engine depending on user interests based on the contents of a web page and the UIH. Fifth, previously studied implicit

indicators for interesting web pages are evaluated. The time spent on a web page and other new indicators are examined in more detail as well.

Experimental results indicate that the personalized ranking methods presented in this study, when used with a popular search engine, can yield more relevant web pages for individual users. The precision/recall analysis showed that our weighted term scoring function could provide more accurate ranking than Google on average.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincerest thanks to Dr. Philip K. Chan, my dissertation advisor, for his encouragement, patience, financial support and greatest guidance through this dissertation, including co-authorship of several papers, which were published as a result of this research. Mere words cannot express my profound appreciation for his endless love and support. I extend thanks to my other committee members, Dr. Debasis Mitra, Dr. Marius-Calin Silaghi, and Dr. Alan C. Leonard.

Appreciation is also acknowledged for the following people who helped me during my dissertation and Ph.D. program:

- Financial support: Dr. Philip K. Chan, Dr. Shirley A. Becker, Dr. Debasis Mitra, Dr. William D. Shoaff;
- Experimental help: Matthew Scripter, Stan Salvador, Matthew Mahoney, Dahee Jung, Timothy, Gaurav Tandon, Rachna Vargiya, Mohammad Arshad, Amanda, Audra, Matt, Turky Alotaiby, Mohsen AlSharif, Akiki, Michel, Ayanna, Jae-gon Park, Ji-hoon, Jun-on, Chris Tanner, Grant Beems;
- Support & Prayer: Young-ki Kim, Matthew Scripter, Nattawut Sridranop, Jae-hyeon Lee, Ji-won Kim, Sun-young Kweon, Seong-won Kim, Gaehlan, Ron, Dana, Kirk, Simon, Paster Luther V. Laite, Paster Warren E. Baker, Dr. Peggy Douglas, Prof. Harry Alston, Se-hoon Kweon, Eun-jeong Lee, Seong-jin Park, Young-chun Bae, Su-bong Ham, Seong-hoon Park, Ali Al-Badi, Marvin Scripter, Faith Scripter, Hun Namgung, Shin-suk Kim, In-suk Gang, Paster Hyoung-woo Park, Paster Hee-youn Lee, etc.;
- Special thanks: Dr. Do-hong Cheon, Dr. Lieberio, Patti Laite.

저의 논문을 위해 희생하신 아버지와 어머니의 노고에 비하면 저는 아무것도 한 것이 없는 것처럼 느껴집니다. 아버지는 새벽기도를 나가시면서 나무를 심으시면서 매일 기도의 마음을 잊지 않으셨습니다. 어머니는 좋아하시는 미역국도 삼가시면서 온 마음의 정성을 다 하셨습니다. 또한 공부는 길고 지루한 과정이기 때문에 습관화가 되지 않으면 그리고 주변의 격려가 없으면 끝까지 견디기가 어렵다고 생각합니다. 항상 연구하는 자세를 보여주시는 아버지의 생활 습관 그리고 항상 적극적으로 믿고 지원해 주신 어머니의 격려가 아니었다면 이 논문을 마치기 힘들었다고 믿습니다. 제 아버님의 성함은 김항남, 어머님은 권성자십니다. 저의 형제들 또한 마치 바위같이 흔들림없이 묵묵히 참으면서 뒤에서 도와준 것에 감사를 드립니다. 형제분들은 김미영, 김혜정, 김정래, 확장된 형제로는 장성수, 스퇴킹어 도미닉, 정래영입니다. 제 작은 아버님들과 어머님들, 그리고 외삼촌분들과 외숙모님들께도 감사 드립니다. 또한 조카들 장문주, 장혁주, 스퇴킹어 막스에게 앞으로 여유있는 외삼촌이 되어 주고 싶습니다. 나이로 인해 이젠 몸의 여러부분이 불편하신 가운데에서도 저의 박사논문을 염려해 주신 외할아버지에게도 감사드립니다. 마지막으로, 연구가 막힐 때 마다 기도를 통해 아이디어를 하나님으로부터 제공받았기에, 한편으론 하나님과 생산적인 대화를 즐겼었다고 봅니다.

Chapter 1

Introduction

Web personalization adapts the information or services provided by a web site to the needs of a user. Web personalization is used mainly in four categories: predicting web navigation, assisting personalization information, personalizing content, and personalizing search results. Predicting web navigation anticipates future requests or provides guidance to client. If a web browser or web server can correctly anticipate the next page that will be visited, the latency of the next request will be greatly reduced (Eirinaki et al., 2004; Kim et al., 2004; Shahabi and Banaei-Kashani, 2003; Cadez et al., 2000). Assisting personalization information helps a user organize his or her own information and increases the usability of the web (Maarek and Ben-Shaul, 1996; Li et al., 1999). Personalizing content focuses on personalizing individual pages, site-sessions (e.g., adding shortcut), or entire browsing sessions (Anderson, 2002). Personalized web search results provide customized results depending on each user's interests (Jeh and Widom, 2003; Haveliwala, 2002; Liu et al., 2003; Bharat and Mihaila, 2001). Information access through a search engine has become an essential part of our daily lives. We use a search engine to find various information from a cloth to technical references. However, the accuracy of search engines is still as low as 55% (Delaney, 2004). In this work, we focus on personalizing web search by ordering search engine results based on the interests of each individual user, which can greatly aid the search through massive amounts of data on the Internet.

1.1. Motivation

When a user browses the web at different times, s/he could be accessing pages that pertain to different topics. For example, a user might be looking for research papers at one time and airfare information for conference travel at another. That is, a user can exhibit different kinds of interests at different times, which provides different contexts underlying a user's behavior. However, different kinds of interests might be motivated by the same kind of interest at a higher abstraction level (computer science research, for example). That is, a user might possess interests at different abstraction levels — the higher-level interests are more general, while the lower-level ones are more specific. During a browsing session, general interests are in the back of one's mind, while specific interests are the current foci. We believe identifying the appropriate context underlying a user's behavior is important in more accurately pinpointing her/his interests. Unlike News Dude (Billsus and Pazzani, 1999), which generates a long-term and a short-term model of interests, we propose to model a continuum of general to specific interests (web browsing interests of a user). The model provides concept hierarchical clusters called a user interest hierarchy (UIH), while suffix tree clustering (STC) algorithm (Zamir and Etzioni, 1998) provides flat clusters.

We can improve the UIH by using phrases in addition to words. A composed term by two or more single words (called “phrase”) usually has more specific meaning and can disambiguate related words. Statistical phrase-finding approaches have been used for expanding vector dimensions in clustering multiple documents (Turpin and Moffat, 1999; Wu and Gunopulos, 2002), or finding more descriptive or important/meaningful phrases (Ahonen et al., 1998; Chan, 1999). We attempt to find more meaningful phrases in a

document. The definition of meaningful is unique to each individual. So, we define a phrase as more meaningful if it is meaningful to the most people.

Building UIH, finding phrases, and devising page-scoring functions (in Chapter 5) can use correlation functions. The analysis of relationships among variables/events is a fundamental task for many data mining problems. There are two types of properties for correlation functions: other properties and desired properties of functions. Other properties such as inversion invariance are depending on each application domain (Tan et al., 2002). One has to examine which properties of other properties are more suitable for his/her application domain (Tan and Kumar, 2000). However, the desirable properties can be applied to all correlation functions. Before selecting or devising a correlation function, it is important to check whether each measure satisfies basic desirable properties of a function (Piatetsky-Shapiro, 1991; Tan et al., 2002). We propose two new desirable properties.

Web personalization adapts the information or services provided by a web site to the needs of a user. There are two main techniques for performing web personalization: collaborative filtering (Eirinaki et al., 2004; Kim et al., 2004; Cadez et al., 2000) and using user profiles (Albanese et al., 2004; Mobasher et al., 1999). Collaborative filtering uses information from many different users to make recommendations. Disadvantages of this method are that it cannot predict a new page and it requires a large amount of data from many users to determine what pages are the most popular. Obtaining data on the web pages visited by many different users is often difficult (or illegal) to collect in many application domains. In contrast, user profiles require the web page history of only a single user. Google (2005) provides personalized services based on explicitly learned user profiles, which consumes users' time and effort. We attempt to personalize the search results using

the user profile that is learned implicitly and uses contents of web pages instead of user's behavior (Kim et al., 2004; Albanese et al., 2004).

A UIH is learned from a set of interesting web pages to a user. Determining a user's interesting web pages can be performed explicitly by asking the user, or implicitly by observing the user's behaviour. Implicit indicators are usually less accurate than explicit indicators (Watson et al., 1998). However, implicit indicators do not require any extra time or effort from the user and can adapt to changes in the user's interests over time. To implicitly measure user interest we need to identify reliable implicit indicators. One of the major user interest indicators identified by researchers is duration, or the time spent on a web page (Granka et al., 2004; Jung, 2001; Claypool et al., 2001; Resnick et al., 1994; Liberman, 1995; Kim et al., 2001; Oard et al., 1998). However, some researches indicate that duration may not be an accurate measure of user interest (Jung, 2001). We suspect that this is because the duration indicator often does not account for the user's absence. We examine the duration in more detail and attempt to propose new other indicators.

1.2. Problem Statement

Our problem consists of three parts as shown in Figure 1: identifying interesting web pages, learning user profile, and personalizing search results. The part of learning user profile can be divided into three sub problems: profile-learning, phrase-finding, and desirable properties of a correlation function. These five main problems for the framework of our web personalization are marked as thick boxes.

The first problem in our system is learning implicit user interest hierarchy for context in personalization. The inputs are the terms (words and phrases) in a set of

interesting web pages or bookmarks. The bookmarked web pages are used when it is difficult to collect a set of interesting web pages to a user. The output is a learned profile called a user interest hierarchy (UIH). We want to devise a method that can learn the UIH from bookmarks implicitly. The second problem is identifying variable-length meaningful phrases with correlation functions from a web page. The inputs are the words in a document. The outputs are meaningful phrases with various lengths. The definition of meaningful is unique to each individual. We let each individual define his or her own definition of meaningful. So, we define a phrase as more meaningful if it is meaningful to the most people. The third problem is about the analysis of desirable properties of correlation functions between two events. These properties of correlation functions help select a right correlation function in learning a user profile, finding meaningful phrases, and scoring interesting web pages. The problem is to find properties that can describe the characteristics of correlation functions. The fourth problem is the personalized ranking of search results with implicitly learned user interest hierarchies. The inputs are a profile (user interest hierarchy) and search results from Google (2005). The outputs will be the reordered search results depending on a user's interests. The goal is to assign higher scores to web pages that a user finds more interesting. The last problem is to find implicit indicators for interesting web pages. The inputs are web log-files and interest scores of web pages provided by a user. The log-files record users' behavior while they are reading web pages. The goal is to find an implicit indicator that could predict the interest score of a web page.

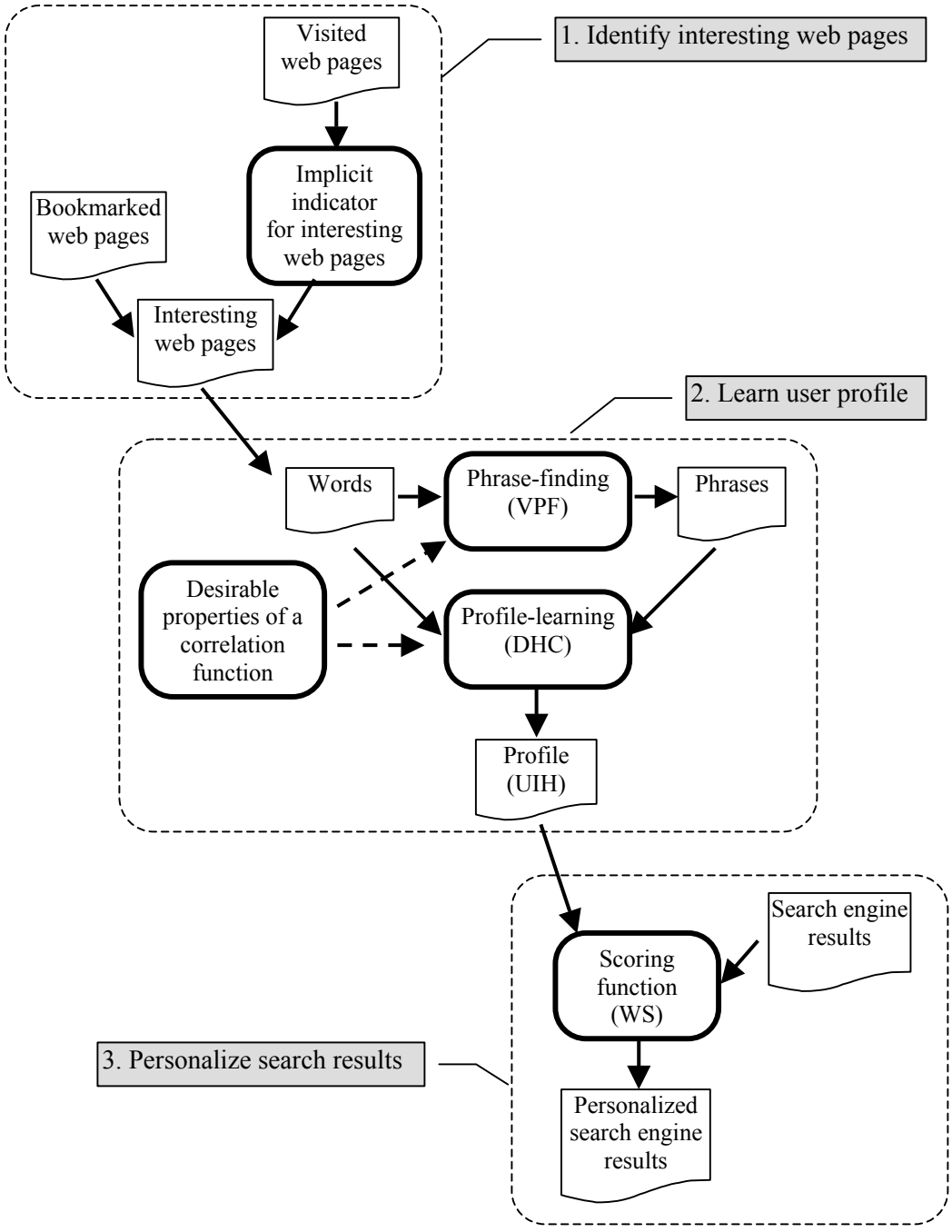


Figure 1. Five research parts in the framework of web personalization

1.3. Approach

The first main objective of this research is to build UIH's that capture general to specific interests without the user's involvement (*implicitly*). The most common and obvious solution for building a UIH is for the user to specify interests explicitly. However, the explicit approach includes these disadvantages: time and effort in specifying interests, and user interest may change over time. Alternatively, an implicit approach can identify a user's interests by inference. Leaf nodes of the UIH generated by our algorithm represent a list of specific user interests. Internal nodes (towards root node) represent more general interests. For example, a graduate student in computer science is looking for a research paper in web personalization. The specific interest is web personalization, but the general interest is computer science. The web pages the student is interested could all be related to computer science and hence words and phrases from these pages would appear in the root node of the UIH. Some of the pages he is interested in could be related to web personalization, and the words (e.g., profile, user, and personalization) might be at the leaf of the UIH. Between the root and the leaves, "internal" tree nodes represent different levels of generality of interest. We devise a divisive hierarchical clustering (DHC) algorithm that constructs such a hierarchy and supports overlapping clusters of the original objects (web pages in our case). Note that clustering web pages is not one of our objectives; the overlapping property allows us to associate a web page to (potentially) multiple topics. We believe our approach has significant benefits and possesses interesting challenges.

Using phrases, in addition to words, we can improve the UIH. A composed term by two or more single words (called "phrase") usually has more specific meaning and can disambiguate related words. For instance, "apple" has different meanings in "apple tree"

and in “apple computer”. Therefore, we propose a variable-length phrase finding algorithm (VPF), which finds more meaningful variable-length phrases. VPF is designed to remove the maximum length of phrases in Ahonen’s algorithm (Ahonen et al., 1998) and fix the problem in Chan’s algorithm (Chan, 1999). VPF increases the length of phrases by adding a word to the phrases made in the previous stage one by one, maintaining high correlation within a phrase. VPF also applies pruning to the phrases collected in order to remove less meaningful phrases.

Both divisive hierarchical clustering (DHC) algorithm and variable-length phrase finding (VPF) algorithm use correlation functions. It is important to understand the characteristics of a correlation function in selecting a correlation function and devising a new correlation function. We analyze the previous desirable properties and propose two new desirable properties. All possible cases that could be made by two events are enumerated and examined. Out of many possible cases, we illustrate that a measure should show a consistent pattern in two cases that are new desirable properties of a correlation function. We believe that these properties will help understand the general characteristics of a measure.

The UIH can be used to build a method for personalizing web search results that is able to reorder the results from Google (2005), such that web pages that a user is most interested in appear at the top of the page. We wish to devise a page scoring function with a user’s implicitly learned User Interest Hierarchy (UIH). Web pages are ranked based on their “score,” where higher scores are considered to be more interesting to the user after comparing the text of the page to the user’s profile. For example, if a user searches for “Australia” and is interested in “travel,” then links related to “travel” will be scored higher;

but if a user is interested in "universities," then pages related to "universities" will be scored higher.

Implicit indicators for interesting web pages help build a UIH implicitly. We compare previous implicit indicators and propose other new implicit indicators. A user's duration on a web page is divided into three types depending on if the browser is open (*complete duration*), if the browser is the active application (*active window duration*), and if the user is looking at the screen (*look at it duration*). We also study new implicit indicators (*memo*) that have not been evaluated in previous research. We divide the web pages visited during our evaluation into two groups: (1) web pages that a user visited more than once and viewed for the longest duration, and (2) all web pages that were visited more than once.

1.4. Key Contributions

The main contributions are:

- we represent user interest hierarchy (UIH) at different abstraction levels (general to specific), which can be learned implicitly from the contents (words/phrases) in a set of interesting pages to a user;
- we build a divisive graph-based hierarchical clustering algorithm (DHC), which constructs a UIH by grouping words (topics) into a hierarchy instead of flat cluster used by STC;
- we propose a variable-length phrase-finding algorithm (VPF) that is designed for finding meaningful phrases – the time complexity remains as $O(n_w)$ where n_w is the number of distinct words in a document;
- more meaningful phrases than previous methods are found and the improvement in performance is statistically significant.

- we identify 2 new desirable properties for a correlation function in general;
- our results indicate that these 2 new desirable properties are more descriptive than the previous 3 desirable properties provided in Piatetsky-Shapiro (1991), because the previous ones highly depend on *cross product ratio* (*CPR*);
- We introduce personalized ranking methods that utilize an implicitly learned user profile (UIH);
- Our experimental results indicate that Weighted Scoring method can achieve higher precision than Google for some top links with *interesting* and *potentially interesting* web pages to the user;
- Our experiments indicate that *complete duration*, *active window duration*, *look at it duration*, and *distance of mouse movement* are reliable indicators for more users than other indicators;

1.5. Dissertation Organization

The rest of this dissertation is organized as follows. In Chapter 2 through 6, we introduce 5 problems of our web personalization architecture. These five parts are learning implicit user interest hierarchy for context in personalization (Kim and Chan, 2003), identifying variable-length meaningful phrases with correlation functions (Kim and Chan, 2004), analysis of desirable properties of correlation functions between two events, personalized ranking of search results with implicitly learned user interest hierarchies, and implicit indicators for interesting web pages. In Chapter 2, we introduce user interest hierarchies (UIH's) and detail our approach towards building an implicit UIH's. Our empirical evaluation regarding the meaningfulness of a UIH is discussed. In Chapter 3, we provide the detailed description of our variable-length phrase-finding algorithm (VPF). Every iteration a word is added to the phrases generated in a previous stage when the correlation value between them is higher than a threshold. This algorithm does not request

any user-defined parameters. The performance of the algorithm will be evaluated by experimental result. In Chapter 4, two new desirable properties of correlation functions are proposed. The experimental comparisons with previous three desirable properties are discussed. In addition to proposing two new properties, we summarize the characteristics of 32 correlation functions based on those properties. In Chapter 5, we incorporate UIHs to the personalization of web search results. Scoring functions are introduced that calculates the *public* and *personal* page score of a web page in the web search results. The function uses four characteristics for a term: the level of a node where a term belongs to (*D*), the length of a term (*L*), the frequency of a term (*F*), and the html formatting used for the emphasis of a term (*E*). The results will be evaluated empirically. In Chapter 6, we learn implicit indicators for interesting web pages to a user and propose more predictable implicit indicators. Some indicators are used frequently and some less frequently. We test both types of implicit indicators empirically. In Chapter 7, we review the areas of web information retrieval, user modeling, and machine learning. In Chapter 8, we summarize our contributions and discuss limitations and future work.

Chapter 2

Learning Implicit User Interest Hierarchy for Context in Personalization

To provide a more robust context for personalization, we desire to extract a continuum of general to specific interests of a user, called a user interest hierarchy (UIH). The higher-level interests are more general, while the lower-level interests are more specific. A UIH can represent a user's interests at different abstraction levels and can be learned from the contents (words/phrases) in a set of web pages bookmarked by a user. We propose a divisive hierarchical clustering (DHC) algorithm to group words (topics) into a hierarchy where more general interests are represented by a larger set of words. Our approach does not need user involvement and learns the UIH "implicitly." To enrich features used in the UIH, we used phrases in addition to words.

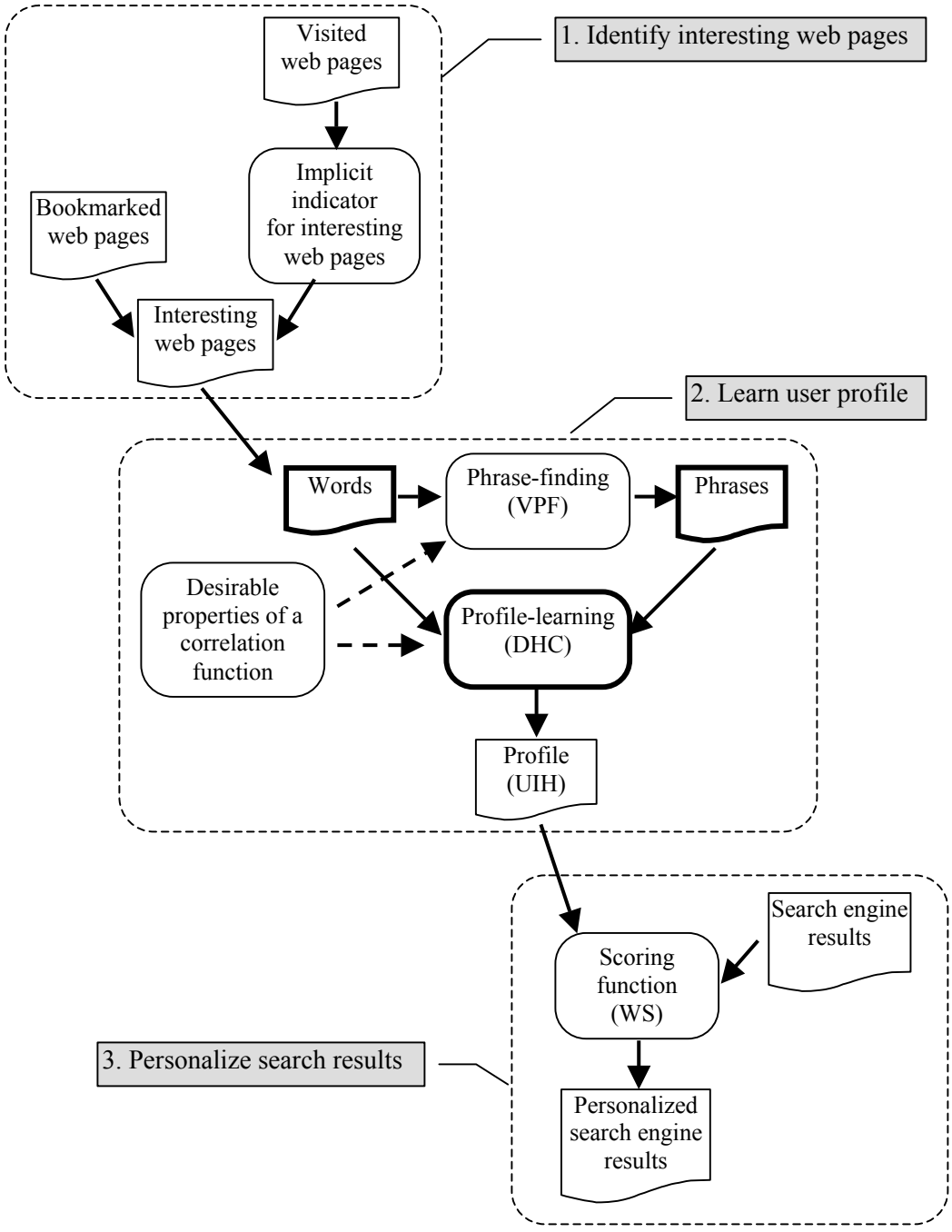


Figure 2. Learning user interest hierarchy in the framework of web personalization

2.1. User Interest Hierarchy

A user interest hierarchy (UIH) organizes a user's general to specific interests. Towards the root of a UIH, more general (passive) interests are represented by larger clusters of words while towards the leaves, more specific (active) interests are represented by smaller clusters of words. To generate a UIH for a user, our clustering algorithm (details in Section 4) accepts a set of web pages bookmarked by the user as input. That is, the input of DHC is web pages that are interesting to a user. We use the words and phrases in the web pages and ignore link or image information. The web pages are stemmed and filtered by ignoring the most common words listed in a stop list (Rasmussen, 1992). The phrases are extracted by VPF algorithm (Kim and Chan, 2004). These processes are depicted in Figure 2.

Table 1. Sample data set

Web	Content
1	ai machine learning ann perceptron
2	ai machine learning ann perceptron
3	ai machine learning decision tree id3 c4.5
4	ai machine learning decision tree id3 c4.5
5	ai machine learning decision tree hypothesis space
6	ai machine learning decision tree hypothesis space
7	ai searching algorithm bfs
8	ai searching algorithm dfs
9	ai searching algorithm constraint reasoning forward
10	ai searching algorithm constraint reasoning forward

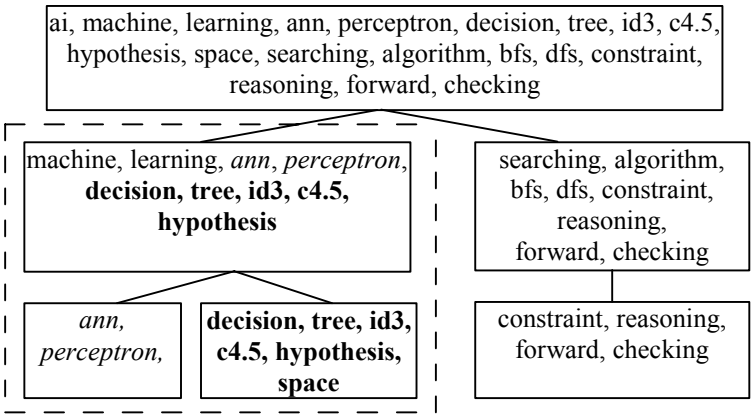


Figure 3. Sample user interest hierarchy

Table 1 contains a sample data set. Numbers on the left represent individual web pages; the content has words stemmed and filtered through the stop list. These words in the web pages can be represented by a UIH as shown in Figure 3. Each cluster, node, can represent a conceptual relationship, for example ‘perceptron’ and ‘ann’ (in italics) can be categorized as belonging to neural network algorithms, whereas ‘id3’ and ‘c4.5’ (in bold) in another node cannot. Words in these two nodes are mutually related to some other words such as ‘machine’ and ‘learning’. This set of mutual words, ‘machine’ and ‘learning’,

performs the role of connecting italic and bold words in sibling clusters and forms the parent cluster. We illustrate this notion in the dashed box in Figure 3.

One can easily identify phrases like “machine learning” and “searching algorithm” in the UIH, however only the individual words are represented in the UIH. By locating phrases from the pages, we can enrich the vocabulary for building the UIH. For example, the phrase “machine learning” can be identified and added to Pages 1-6. If we can use phrases as feature in the UIH, each cluster will be enriched because phrases are more specific than words. For example, a user is interested in both phrases “java coffee” and “java language”. The word “java” will be in the parent cluster of both “coffee” and “language”. Each child cluster would contain only “coffee” or “language”, which is relatively less useful when not in combination with “java”.

The approach we take to generate the hierarchy is similar to clustering pages, but pages may belong to multiple clusters – overlapping clusters of pages. That is, instead of directly clustering the original objects (web pages), we first cluster features (words) of the objects and then the objects are assigned to clusters based on the features in each cluster. Note that a document can have terms in different clusters, hence, a document can be in more than one cluster. Since the more challenging step is the initial hierarchical clustering of features, our primary focus for this Chapter is on devising and evaluating algorithms for this step.

2.2. Building User Interest Hierarchy

We desire to learn a hierarchy of interest topics from a user's web pages bookmarked by a user, in order to provide a context for personalization. Our divisive hierarchical clustering (DHC) algorithm recursively partitions the words into smaller clusters, which represent more related words. We assume words occurring close to each other (within a window size) are related to each other. We investigate correlation functions that measure how closely two words are related in Section 4.2. We also investigate techniques that dynamically locate a threshold that decides whether two words are strongly related or not in Section 4.3. If two words are determined to be strongly related to each other, they will be in the same cluster; otherwise, they will be in different clusters.

2.2.1. Algorithm

Our algorithm is a divisive graph-based hierarchical clustering method called DHC, that recursively divides clusters into child clusters until it meets the stopping conditions. In preparation for our clustering algorithm, we extract words from web pages that are interesting to the user, filter them through a stop list, and stem them (Rasmussen, 1992). Figure 4 illustrates the pseudo code for the DHC algorithm. Using a correlation function, we calculate the strength of the relationship between a pair of words in line 1. After calculating a threshold to differentiate strong correlation values from weak correlation in line 2, we remove all weak correlation values in line 5. We then build a weighted undirected graph with each vertex representing a word and each weight denoting the correlation between two words. Since related words are more likely to appear in the

same document than unrelated terms, we measure co-occurrence of words in a document. Given the graph, called a `CorrelationMatrix`, the clustering algorithm recursively partitions the graph into sub-graphs, called Clusters, each of which represents a sibling node in the resulting UIH in line 6.

At each partitioning step, edges with "weak" weights are removed and the resulting connected components constitute sibling clusters (we can also consider cliques as clusters, but more computation is required). We treat the determination of what value is considered to be "strong" or "weak", as another clustering. The recursive partitioning process stops when one of the stopping criteria is satisfied. The first criterion is when the current graph does not have any connected components after weak edges are removed. The second criterion is a new child cluster is not formed if the number of words in the cluster falls below a predetermined threshold.

```
Cluster: distinct words in a set of interesting web pages to a user
    [with information of web page membership]
CORRELATIONFUNCTION: Calculates the "closeness" of two words.
FINDTHRESHOLD: Calculates the cutoff value for determining strong
    and weak correlation values.
WindowSize: The maximum distance (in number of words) between two
    related words in calculating their correlation value.

Procedure DHC (Cluster,CORRELATIONFUNCTION,FINDTHRESHOLD,WindowSize)
1.  CorrelationMatrix ← CalculateCorrelationMatrix
        (CORRELATIONFUNCTION, Cluster, WindowSize)
2.  Threshold ← CalculateThreshold(FINDTHRESHOLD, CorrelationMatrix)
3.  If all correlation values are the same or a threshold is not
    found
4.     Return EmptyHierarchy
5.  Remove weights that are less than Threshold from
    CorrelationMatrix
6.  While (ChildCluster←NextConnectedComponent (CorrelationMatrix))
7.     If size of ChildCluster >= MinClusterSize
8.        ClusterHierarchy←ClusterHierarchy + ChildCluster +
             DHC(ChildCluster,CORRELATIONFUNCTION,FINDTHRESHOLD,
    WindowSize)
9. Return ClusterHierarchy
End Procedure
```

Figure 4. DHC algorithm

Suppose we built a weighted undirected graph with the running example in Table 1 where each vertex represents a word and each weight (value) denotes the correlation value. The undirected graph can be depicted as shown in a) in Figure 5 and Figure 6 – the left column shows graph partitioning and the right column represents the corresponding tree. We presented only some vertices and edges as shown in a). Once a threshold for differentiating "strong" edges from "weak" edges is calculated by using a `Findthreshold` method, we can remove weak edges – those removed edges are represented as dashed lines. After removing weak edges, DHC finds connected components, which is shown in b). If the number of elements in a cluster is greater than the minimum number of elements in a cluster (e.g., 4), then the correlation values are recalculated and the algorithm repeats the process of removing "weak" edges as shown in c). Since DHC recursively partitions the graph into subgraphs, called Clusters, the final result becomes hierarchical clusters as shown in d).

The `CalculateCorrelationMatrix` function takes a correlation function, cluster, and window size as parameters and returns the correlation matrix, where the window size affects how far two words (the number of words between two words) can be considered as related. The `CalculateThreshold` function takes a threshold-finding method and correlation matrix as parameters and returns the threshold. The correlation function (Sec. 4.2) and threshold-finding method (Sec. 4.3) greatly influence the clustering algorithm, and are discussed next. .

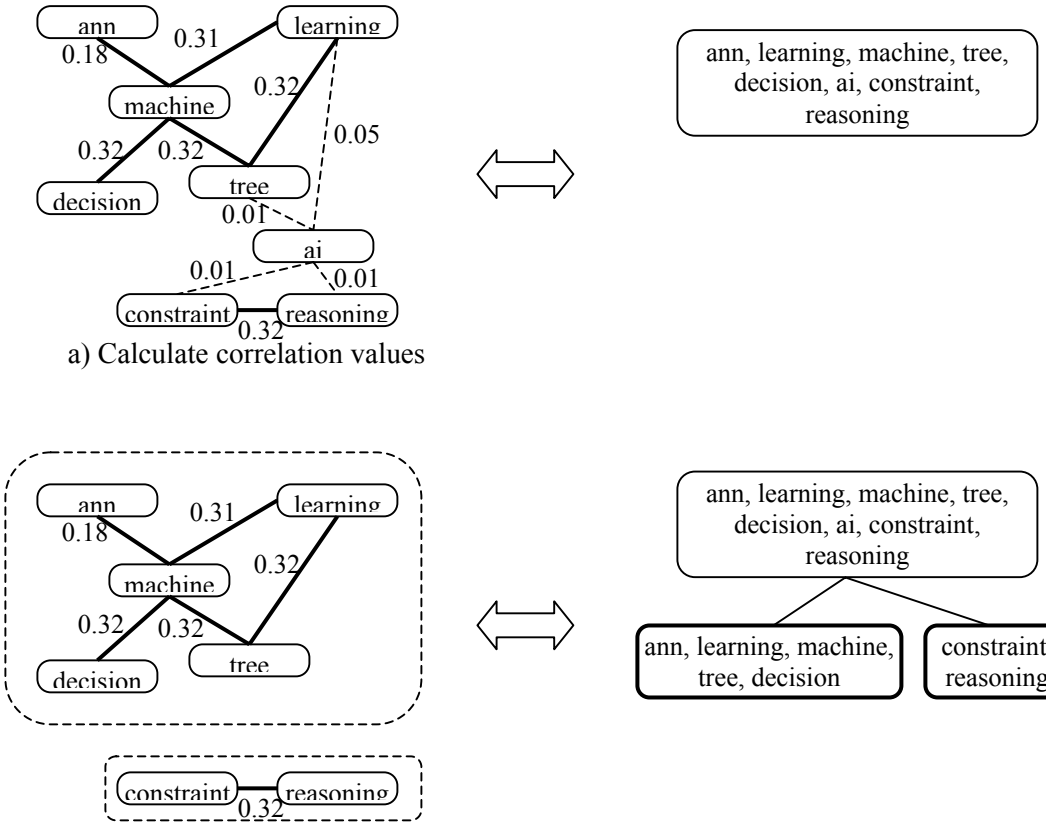
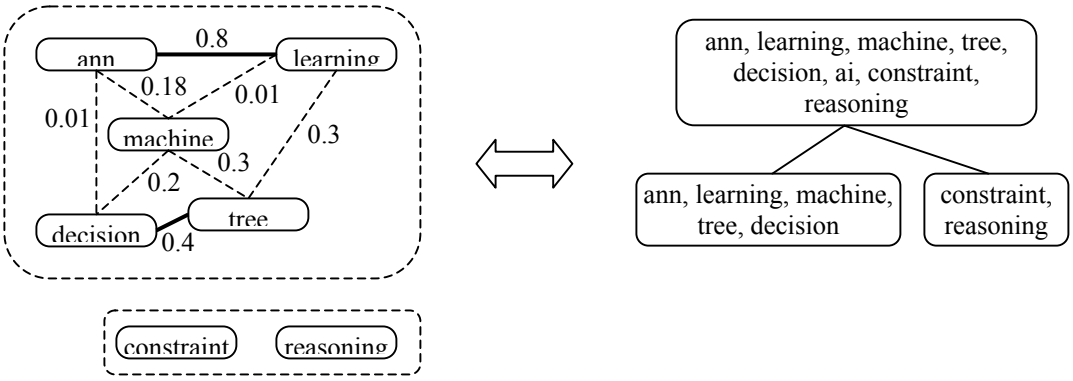
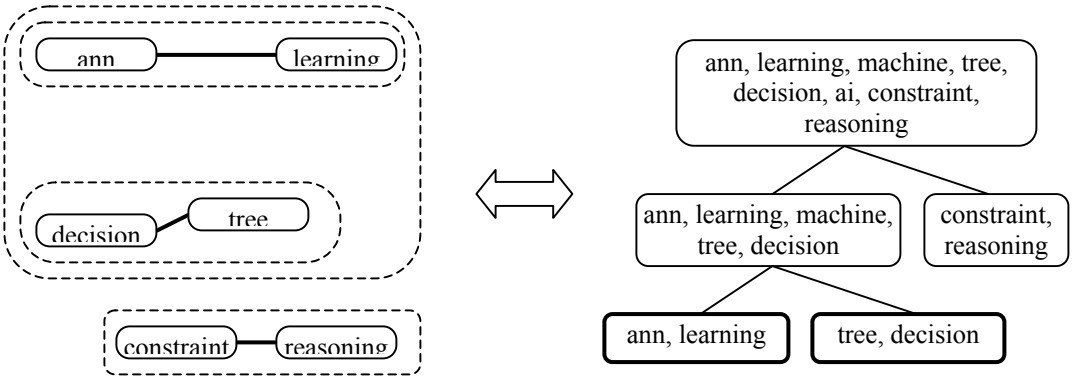


Figure 5. An example of DHC algorithm (1)



c) Recalculate correlation values



d) Find connected components

Figure 6. An example of DHC algorithm (2)

2.2.2. Correlation Functions

The correlation function calculates how strongly two words are related. Since related words are likely to be closer to each other than unrelated words, we assume two words co-occurring within a window size are related to each other. To simplify our discussion, we have been assuming the window size to be the entire length of a document. That is, two words co-occur if they are in the same document. These functions are used in `CalculateCorrelationMatrix` function in Figure 4.

2.2.2.1. AEMI

We use AEMI (Augmented Expected Mutual Information) (Chan, 1999) as a correlation function. AEMI is an enhanced version of MI (Mutual Information) and EMI (Expected Mutual Information). Unlike MI which considers only one corner of the contingency matrix and EMI which sums the MI of all four corners of the contingency matrix, AEMI sums supporting evidence and subtracts counter-evidence. Chan (1999) demonstrates that AEMI could find more meaningful multi-word phrases than MI or EMI. Concretely, consider A and B in AEMI (*A*,*B*) are the events for the two words. $P(A = a)$ is the probability of a document containing a and $P(A = \bar{a})$ is the probability of a document not having term a. $P(B = b)$ and $P(B = \bar{b})$ is defined likewise. $P(A = a, B = b)$ is the probability of a document containing both terms *a* and *b*. These probabilities are estimated from documents that are interesting to the user. AEMI (*A*,*B*) is defined as:

$$AEMI(A,B) = P(a,b)\log\frac{P(a,b)}{P(a)P(b)} - \sum_{(A=a,B=\bar{b})(A=\bar{a},B=b)} P(A,B)\log\frac{P(A,B)}{P(A)P(B)} \qquad \text{Eq. 1}$$

The first term computes supporting evidence that *a* and *b* are related and the second term calculates counter-evidence. Using our running example in Figure 3, Table 2 shows a few examples of how AEMI values are computed. The AEMI value between 'searching' and 'algorithm' is 0.36, which is higher than the AEMI value between 'space' and 'constraint', –0.09.

Table 2. AEMI values

$P(a)$	$P(\bar{a})$	$P(b)$	$P(\bar{b})$	$P(ab)$	$P(\bar{a}b)$	$P(a\bar{b})$	$AEMI(a,b)$
a = searching, b = algorithm							
0.4	0.6	0.4	0.6	0.4	0	0	0.36
a = space, b = constraint							
0.2	0.8	0.2	0.8	0	0.2	0.6	-0.09
a = ann, b = perceptron							
0.2	0.8	0.2	0.8	0.2	0	0	0.32

2.2.2.2. AEMI-SP

Inspired by work in the information retrieval community, we enhance AEMI by incorporating a component for inverse document frequency (IDF) in the correlation function. The document frequency of a word calculates the number of documents that contain the word. Words that are commonly used in many documents are usually not informative in characterizing the content of the documents. Hence, the inverse document frequency (the reciprocal of document frequency) measures how informative a word is in characterizing the content. Since our formulation is more sophisticated than IDF and it involves a pair of words rather than one word in IDF, we use a different name and call our function specificity (SP).

We estimate the probability of word occurrence in documents instead of just document frequency so that we can scale the quantity between 0 and 1. We desire to give high SP values to words with a probability below 0.3 (approximately), gradually decreasing values from 0.3 to 0.7, and low values above 0.7. This behavior can be approximated by a sigmoid function, commonly used as a smoother threshold function in neural networks, though ours needs to be smoother. SP(x) is defined as: $1/(1 + exp(0.6 \times (x \times 10.5 - 5)))$, where x is defined as: MAX ($P(a)$, $P(b)$) - we choose the larger probability so that SP is more conservative. The factor 0.6 smoothes the curve, and constants 10.5 and –5 shift the range of x from between 0 and 1 to between -5 and 5.5. The new range of -5 and 5.5 is slightly asymmetrical because we would like to give a small bias to more specific words. For instance, for a = 'ann' and b = 'perceptron', x is 0.2 and SP(x) is 0.85, but for a='machin' and b='ann', x is 0.6 and SP(x) is 0.31.

Our correlation function AEMI-SP is defined as: AEMI × SP/2. The usual range for AEMI is 0.1–0.45 and SP is 0–1. To scale SP to a similar range as AEMI, we divide SP by 2. For example, in Table 3 the AEMI-SP value for 'searching' and 'algorithm' is lower than the value for 'ann' and 'perceptron' because the SP value for 'ann' and 'percetpron' is higher even though the AEMI value is lower.

Table 3. AEMI-SP values

	AEMI	**SP**	**AEMI-SP**
a = searching ***b = algorithm***	0.36	0.62	0.113
a = ann ***b = perceptron***	0.32	0.85	0.137

2.2.2.3. Other Correlation Functions

We also investigated other existing correlation functions. The *Jaccard* function (Rasmussen, 1992) is defined as: $\frac{P(a,b)}{P(a \cup b)}$. When a word describes a more general topic, we expect it to occur quite often and appear with different, more specific words. Hence, we desire general ("connecting") words to exist only at higher levels in the UIH. For example, 'ai' is general and preferably should not appear at the lower levels. Using our running example in Figure 3, the *Jaccard* value between 'ai' and 'machine' is 0.6 and the value between 'ai' and 'search' is 0.5. If the threshold is 0.49, both pairs are in the same cluster and 'ai' may perform the role to connect 'machine' and 'search'. Even if the threshold is 0.55, 'ai' still remains in the child cluster with 'machine' (since their correlation value is over the threshold) - which is a wrong decision. This phenomenon tells us the *Jaccard* function is not proper for making hierarchical clusters.

The *MIN* method in STC (Zamir and Etzioni, 1998) can be defined as *MIN*($P(a|b)$, $P(b|a)$). The idea is that if we assign the same correlation value to connected words and connecting words, they would go together. For instance, 'ai' connects 'machine' and 'searching', so they are grouped together in one cluster. However, when they are divided into child clusters, 'ai' should be removed because 'ai' is too general. But *MIN* (P('ai'|'machine'), P('machine'|'ai')) still yields a relatively higher value than the average. Alternatively, the MAX function, *MAX*($P(a|b)$,$P(b|a)$), does not distinguish the value for 'ai' and 'machine', and the value for 'machine' and 'learning', even though the latter pair has a much stronger relationship. Since *Jaccard*, *MIN*, and *MAX* did not generate desirable cluster hierarchies, we excluded them from further experiments.

2.2.3. Threshold-finding Methods

Instead of using a fixed user-provided threshold (as in STC (Zamir and Etzioni, 1998)) to differentiate strong from weak correlation values between a pair of words, we examine methods that dynamically determine a reasonable threshold value. Weights with a weak correlation are removed from `CorrelationMatrix` and child clusters are identified.

Table 4. Distribution of frequency and number of children

Region	Range	Freq.	# of Children
0	0.27 <= x < 0.28	16	Not counted
1	0.28 <= x < 0.29	0	Not counted
2	0.29 <= x < 0.30	0	Not counted
3	0.30 <= x < 0.31	1	Not counted
4	0.31 <= x < 0.32	0	Not counted
5	0.32 <= x < 0.33	13	6
6	0.33 <= x < 0.34	0	1
7	0.34 <= x < 0.35	0	1
8	0.35 <= x < 0.36	0	1
9	0.36 <= x	2	Not applicable

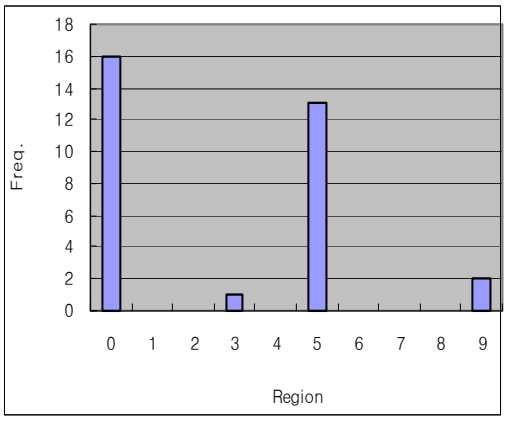


Figure 7. Shown in a Histogram

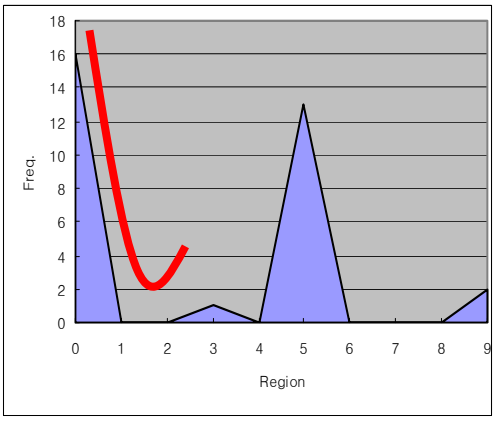


Figure 8. Find the widest and deepest valley

2.2.3.1. Valley

To determine the threshold, we would like to find a sparse region that does not have a lot of similar values. That is, the frequency of weights in that region is low. We first determine the highest observed and lowest desirable correlation values, and quantize the interval into ten regions of equal width. The lowest desirable correlation value is defined as the value achieved by a pair of words that occur together only in one document. We then determine the frequency of values in each region. Generally, lower weights have a higher frequency and higher weights have a lower frequency. If the frequency monotonically decreases with regions of higher weights, picking the region with the lowest frequency will always be the region with the highest weights. If, unfortunately, the threshold is too high, then too many edges will be cut. In this case, the threshold is set to be the average plus standard deviation (biasing to remove more edges with lower weights).

However, if the frequency does not decrease monotonically, we attempt to identify the "widest and steepest" valley. A valley is defined as any region where the frequency decreases and then increases. Steepness can be measured by the slopes of the two sides of a valley and the width of how many regions the valley covers. Since the regions are of equal width, we calculate "quality" of a valley by: $\sum_{i,j} \left| freq_i - freq_j \right|$, where i and j are successive regions on the two sides of a valley. Once the widest and steepest valley is located, we identify the threshold in the region that constitutes the bottom (lowest frequency) of the valley.

For example, in Table 4, the first column is the id of each region, the second column is the range of correlation values, the third column is the number of values resides in each region, and the last column is the number of child nodes that can be generated with

the lowest value in each range as a threshold. There are three valleys when a histogram is drawn like Figure 7: one from Region 0 through 3, (quality is 17), another one from Region 3 through 5, (quality is 14), and the last one is from Region 5 through 9, (quality is 15). Therefore, the widest and steepest valley is the first valley and its bottom is in Regions 1 and 2, which is shown in Figure 8. To identify the threshold inside the bottom region, we ignore the frequency information and find two clusters of correlation values. In this case, it is a one-dimensional two-cluster task, which can be accomplished by sorting the weights and splitting at the largest gap between two successive weights (Largest gap). In our example, since the bottom has zero frequency, any value between .28 and .30 can be the threshold. If the bottom does not have zero frequency, we recursively divide the bottom until the frequency is zero.

2.2.3.2. MaxChildren

The MaxChildren method selects a threshold such that maximum of child clusters are generated. This way we divide the strongly correlated values from weakly correlated ones. This also ensures that the resulting hierarchy does not degenerate to a tall and thin tree (which might be the case for other methods). This preference also stems from the fact that topics are generally more diverse than detailed and the library catalog taxonomy is typically short and wide. For example, we want the trees in Figure 3 to be shorter and wider. MaxChildren calculates the number of child clusters for each boundary value between two quantized regions. To guarantee the selected threshold is not too low, this method ignores the first half of the boundary values. For example, in Table 4, the boundary value 0.33 (between Regions 5 and 6) generates the most children and is selected as the

threshold. This method recursively divides the selected best region until there are no changes on the number of child clusters.

2.2.3.3. Other Threshold-finding Methods

There are some other threshold-finding methods that we initially studied but found to be inferior to Valley or MaxChildren, and subsequently are not included in this Chapter. LargestGap sorts the values and split at the largest gap between two successive values (this method can be used in the Valley method after the bottom of the largest valley is found). Again this is motivated by trying to form two clusters in a one-dimensional space. However, in our initial experiments, the largest gap is close to the largest observed value and thus the resulting tree is usually too small. To prevent the threshold from being too large, Top30% method selects a threshold that retains values in the top 30%. However, this method generates tall and thin trees. To retain 'abnormally' large values of a threshold, we also studied Average+StandardDeviation, in order to select a threshold larger than the average. This is later combined into the Valley method.

2.2.4. Window Size and Minimum Size of a Cluster

The window size parameter specifies the maximum 'physical' distance (in terms of number of words) between a pair of words for consideration of co-occurrence. We have been using the entire document length as the window size to simplify our discussion. However, considering two words occurring in the same page as related might be too optimistic. Hence, we investigated smaller window sizes that roughly cover a paragraph (e.g., 100 words) or a sentence (e.g., 15 words). However, in our experiments the window

size does not make a significant difference. And, the minimum size of a cluster affects the number of clusters. A larger number of clusters makes the hierarchy less comprehensible and requires more computation. We picked 4 as the minimum size of a cluster, because this number of words can represent a concept that is sufficiently specific.

2.3. Experiments

We will evaluate the UIH itself to see if it is meaningful using real data. The quality of the UIH will also describe the performance of DHC. We then compared user interest hierarchies using different methods. Furthermore, we compared the quality of UIHs of which one uses words only and the other includes phrases.

2.3.1. Experimental Data and Procedures

Experiments were conducted on data obtained from our departmental web server. By analyzing the server access log from January to April 1999, we identified hosts that were accessed at least 50 times in the first two months and also in the next two months. We filtered out proxy, crawler, and our computer lab hosts, and identified "single-user" hosts, which are at dormitory rooms and a local company (Chan, 1999). We yielded 13 different users and collected the web pages they visited. The total pages that we used were around 1400 files and most of the pages contained regular contents such as no media biased files. The number of words on the web pages was on the average 1,918 – minimum number of words was 340, and maximum was 3,708. To find phrases we used variable-length phrase finding algorithm (VPF) (Kim and Chan, 2004) because it finds more meaningful phrases

than other methods (Chan, 1999; Ahonen et al., 1998). We evaluated the effectiveness of our algorithms by analyzing the generated hierarchies in terms of meaningfulness and shape. Separate experiments were conducted to evaluate the effectiveness of different correlation functions, threshold-finding methods, and window sizes. In order to remove the researcher's bias, we randomly reordered whole clusters from all approaches, before we evaluate each cluster.

2.3.2. Evaluation Criteria

To evaluate a UIH, we use both qualitative and quantitative measures. Qualitatively, we examine if the cluster hierarchies reasonably describe some topics (meaningfulness). Quantitatively, we measure shape of the cluster trees by calculating the average branching factor (ABF) (Russell and Norvig, 1995). ABF is defined as the total number of branches of all non-leaf nodes divided by the number of non-leaf nodes.

We categorized meaningfulness as 'good', 'bad,' or 'other'. Since the leaf clusters should have specific meaning and non-leaf clusters are hard to interpret due to their size, we only evaluated the leaf clusters for meaningfulness. Our measure is based on interpretability and usability (Han, 2001). So, we check two properties of the leaf clusters: the existence of related words, and possibility of combining words. For instance, for related words, consider 'formal', 'compil', 'befor', 'graphic', 'mathemat', and 'taken' are in a cluster, even though 'befor' and 'taken' do not have any relationship with the other words. Since the words, 'formal', 'compil', 'graphic', and 'mathemat', are classified as class names related to computer science major, this cluster is evaluated as 'good'. For the possibility of combining words, consider 'research', 'activ', 'class', and 'web' are in a

cluster. In this case, the meaning of the cluster can be estimated as ‘research activity’ or ‘research class’ (Zamir and Etzioni, 1999), so we regard this cluster as ‘good’. A cluster is marked as ‘good’ when it has more than 2/5 of the words that are related or have more than 2 possible composite phrases as well. This is hard to measure, so we attempted to be as skeptical as possible. For example, suppose a cluster has ‘test’, ‘info’, ‘thursdai’, ‘pleas’, ‘cours’, ‘avail’, and ‘appear’. In this case one can say ‘test info’ or ‘cours info’ are possible composite phrases, but ‘test info’ does not have any conceptual meaning in our opinion, so we did not count that phrase. If a cluster contains less then 15 words and does not satisfy the criteria for ‘good’ cluster, it is marked as ‘bad’. A cluster is marked as ‘other’ when a leaf cluster has more than 15 words because a big leaf cluster is hard to interpret.

We categorized shape as ‘thin’, ‘medium,’ or ‘fat’. If a tree’s ABF value is 1, the tree is considered a ‘thin’ tree (marked as ‘T’ in the following tables). If the ABF value of a tree is at least 10, the tree is considered a ‘fat’ tree (marked as ‘F’). The rest are ‘medium’ trees (marked as ‘M’). We consider one more tree type: ‘conceptual’ tree (marked as ‘C’), which subsumes ‘M’ or ‘F’ type trees. A conceptual tree is one that has at least one node with more than two child clusters and more than 80% of the words in each child cluster have similar meaning. Since we prefer a tree that can represent meaningful concepts, ‘C’ type trees are the most desirable. ‘T’ type trees are degenerate (imagine each node in the hierarchy has only one child and the hierarchy resembles a list, which is usually not how concepts are organized) and hence undesirable. Based on these evaluation criteria, we analyze different correlation functions, threshold-finding methods and window sizes.

2.4. Results and Analysis

In this section we analyze the results from the DHC. We first evaluate the DHC algorithm with only words as features. Then, we compare the results from DHC using only words and the combination of words and phrases as features.

2.4.1. Building UIH with Only Words as Features

2.4.1.1. Correlation Functions

We compared two correlation functions: AEMI versus AEMI-SP. We fixed the threshold-finding method to Valley and the window size to 'entire page.' Table 5 and Table 6 illustrate the results. The letter 'U' stands for user, '# of L' means the number of leaf nodes. 'G %' means 'percentage of good', which is calculated by dividing the number of 'good' leaves by the '# of L'. AEMI yielded significantly more meaningful leaf clusters (59% good) than AEMI-SP (41% good). The means of the two groups were significantly different from each other according to the t-test at level 0.05 (Lind et al., 2002). Both methods generated trees whose shapes were mostly 'medium'. For U8, AEMI generated a conceptually related tree — the tree had a node with two child clusters, which contained words from course titles and hence represented the concepts of different classes. For U2 with AEMI-SP, the generated tree was 'fat' and had an ABF value of 10.

2.4.1.2. Threshold-finding Method

We compared two threshold-finding methods: Valley versus MaxChildren. We fixed the correlation function to AEMI and the window size to entire page. Table 5 and Table 7 illustrate the results. MaxChildren generated more meaningful leaf clusters (59% good) than Valley (47% good). However, the means of two groups were not statistically different from each other according to the t-test at level 0.05. Tree shapes are similar (medium) in both methods. However, generally, trees generated by MaxChildren were shorter, which indicates that MaxChildren reduces the number of iterations in the DHC algorithm by dividing the cluster in an early stage. Hence, MaxChildren is faster than Valley.

2.4.1.3. Window Size

We compared the performance using different window sizes: 'entire page' versus 100 words (paragraph length). We fixed the correlation function to AEMI and the threshold-finding method to MaxChildren. Table 5 and Table 8 illustrate the results. A window size of the entire page generated slightly more meaningful clusters (59% good) than a window size of 100 (57% good). However, a window size of 100 yielded more tress with 100% 'good' leaf clusters (6) than a window size of the entire page (5). Hence, it is not clear which window size produces more meaningful clusters. Both methods resulted in 'medium' trees. A window size of 100 generated one thin tree for U11. The 'T' tree in Table 8 has only two nodes: the root and one leaf.

Table 5. Combination of AEMI, MaxChildren, and entire page

User	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	U13	Sum
# of L	4	4	3	6	4	4	2	6	4	8	8	4	2	59
Good	3	2	2	5	3	2	2	6	3	2	1	3	1	35
Bad	1	2	1	1	1	2			1	6	7	1	1	24
Other														0
G %	75	50	67	83	75	50	100	100	75	25	13	75	50	**59**
ABF	2.5	2	2	2.7	2	2	2	2.2	2.5	2.4	2.4	2.5	2	
Shape	M	M	M	M	M	M	M	C	M	M	M	M	M	

Table 6. Combination of AEMI-SP, MaxChildren, and entire page

User	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	U13	Sum
# of L	10	10	5	10	9	7	7	5	10	13	17	8	4	115
Good	2	6	1	3	3	3	3	3	4	5	6	4	4	47
Bad	8	4	4	7	6	4	2	2	4	5	8	4		58
Other							2		2	3	3			10
G %	20	60	20	30	33	43	43	60	40	38	35	50	100	**41**
ABF	2.8	10	2.3	3.3	3	3	2.5	3	4	2.7	2.8	3.3	2.5	
Shape	M	F	M	M	M	M	M	M	M	M	M	M	M	

Table 7. Combination of AEMI, Valley, and entire page

User	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	U13	Sum
# of L	6	6	4	6	5	5	4	3	3	8	11	4	7	72
Good	4	4	1	5	2	3	4	1	1	1	2	3	3	34
Bad	2	1	3	1	2	2		2	2	7	7	1	4	34
Other		1			1						2			4
G %	67	67	25	83	40	60	100	33	33	13	18	75	43	**47**
ABF	2.7	2	2	2.7	2.3	2.3	2	2	3	2.5	2.4	2.5	2.5	
Shape	M	M	M	M	M	M	M	M	M	M	M	M	M	

Table 8. Combination of AEMI, MaxChildren, and 100 words

User	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	U13	Sum
# of L	5	2	12	9	4	4	2	7	8	13	1	6	4	77
Good	5	2	3	5	4	3	2	7	3	2	1	3	4	44
Bad			8	4		1			5	11		3		32
Other			1											1
G %	100	100	25	56	100	75	100	100	38	15	100	50	100	**57**
ABF	3	2	4.7	3.7	2.5	2.5	2	3	3.3	3.4	1	3.5	4	
Shape	M	M	M	M	M	M	M	M	M	M	T	M	M	

2.4.2. Building UIH with Words and Phrases as Features

If we can add phrases as feature in the UIH, each cluster will be enriched because phrases are more specific than words. We compared two different data sets: one consisting of only words and the other consisting of words and phrases – the phrases were collected by VPF (Kim and Chan, 2004). Table 5 and Table 9 illustrate the results. Results from the data with phrases presented more meaningful leaf clusters (64%) than results with only words (59%). Tree shapes were similar (medium) in both methods.

UIHs learned from a user (U1) are depicted in Figure 9 and Figure 10. The one from only words has three 'good' leaf clusters (1, 3, and 4) and one 'bad' leaf cluster (5). Cluster 0 denotes root nodes, which has all words or phrases. The right one which is learned from both words and phrases has all 'good' clusters; furthermore, it is more descriptive because Clusters 6 and 7 in Figure 10 describe class names while cluster 3 in Figure 9 alone describes class names. We can say Clusters 6 and 7 in Figure 10 are conceptually related because both are class names. We cannot explain why some specific interests in one UIH do not exist in the other UIH. For example, Clusters 4 in Figure 9 showed that the user (U1) was interested in a Master or Doctoral degree program, but the interest in Master degree did not exist in the UIH in Figure 10.

Table 9. Use words and phrases

User	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10	U11	U12	U13	Sum
# of L	6	2	13	8	4	5	3	10	8	15	1	6	4	85
Good	6	2	3	4	2	4	3	10	5	8	1	2	4	54
Bad			9	4	2	1			3	7		4		30
Other			1											1
G %	100	100	23	50	50	80	100	100	63	53	100	33	100	**64**
ABF	3.5	2	5	3.4	2.5	3	2	5.5	3.4	3.8	1	3.5	4	
Shape	C	M	M	M	M	M	M	C	M	M	T	M	M	

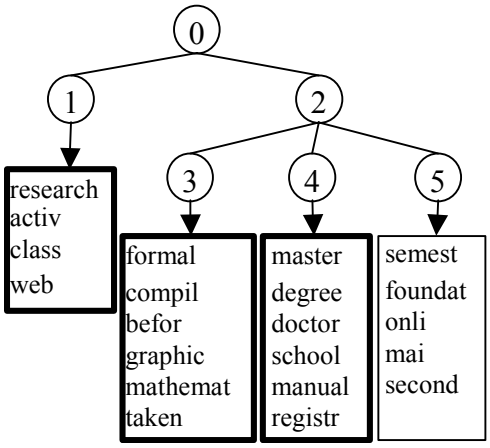


Figure 9. UIH with words

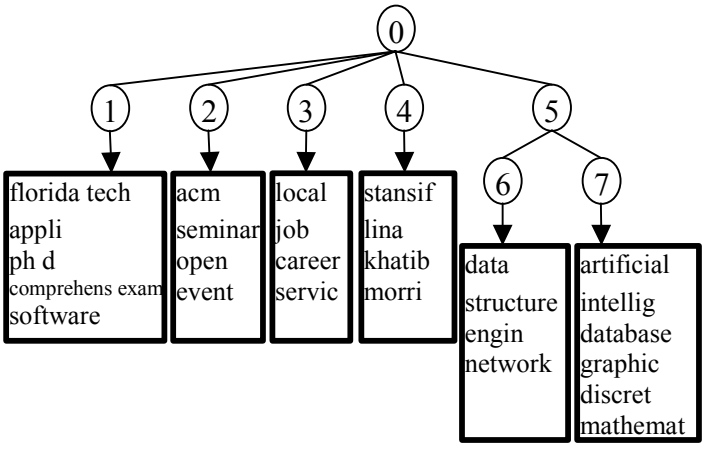


Figure 10. UIH with words and phrases

2.5. Summary

To create a context for personalization, we proposed establishing a user interest hierarchy (UIH) that can represent a continuum of general to specific interests from a set of web pages interesting to a user. This approach is non-intrusive and allows web pages to be associated with multiple clusters/topics. We proposed our divisive hierarchical clustering (DHC) algorithm and evaluated it based on data obtained from 13 users on our web server. We also introduced correlation functions and threshold-finding methods for the clustering algorithm. Our empirical results suggested that DHC with the AEMI correlation function and the MaxChildren threshold-finding method yielded more meaningful leaf clusters. In addition, by using phrases found by VPF algorithm, we improved performance up to 64% of interpretable clusters. We did not analyze differences among the UIHs' obtained from various users because of the large numbers of web pages used in our experiments. Results from experiments not reported here indicated that stemmed words were more effective than whole words. The minimum cluster size affected the number of leaf clusters; size 4 was easy to use and seemed to produce reasonable results.

The performance of the DHC algorithm varied depending on a user and the articles selected. We currently do not understand the reason for the variance in performance. We assume this is due to intrinsic characteristics of a user and an article.

Chapter 3

Identifying Variable-Length Meaningful Phrases with Correlation Functions

Finding phrases in a document has been studied in various information retrieval systems to improve their performance. Many previous statistical phrase-finding methods had a different aim such as document classification. Some are hybridized with statistical and syntactic grammatical methods; others use correlation heuristics between words. We propose a new phrase-finding algorithm that adds correlated words one by one to the phrases found in the previous stage, maintaining high correlation within a phrase. The inputs are words in a document and the outputs are phrases collected. These processes are depicted in Figure 11 with thicker features.

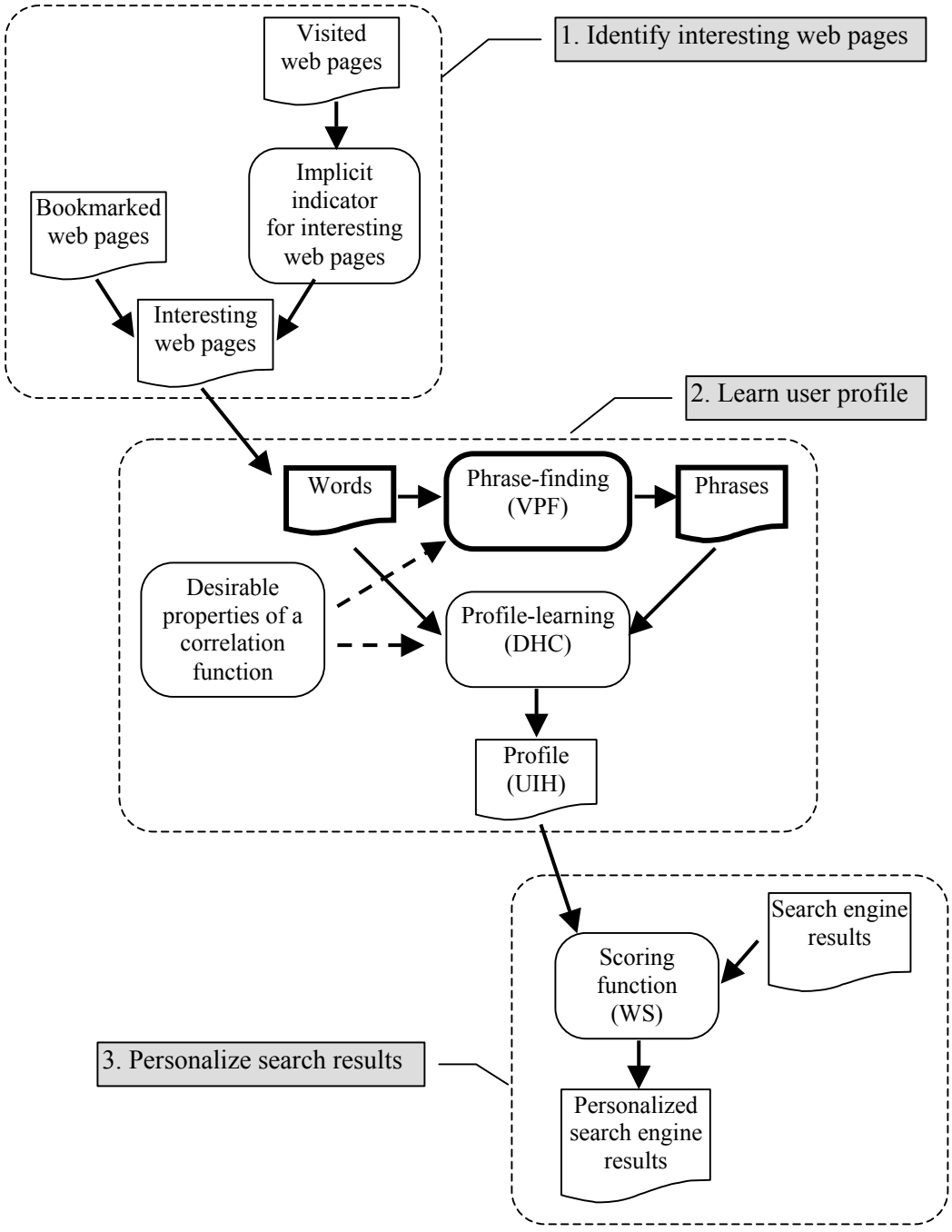


Figure 11. Finding phrases in the framework of web personalization

3.1. Variable-length Phrases

Our algorithm consists of two components: the main algorithm and the correlation function. In preparation for VPF, we extract words from a web page visited by the user, filter them through a stop list, and stem them (Ahonen et al., 1998; Chan, 1999; Frakes and Baeza-Yates, 1992; Zamir and Etzioni, 1998). Other phrase-finding algorithms also require these pre-processing steps.

3.1.1. VPF Algorithm

Our algorithm adds words one by one to the phrases found in a previous stage – each stage corresponds to each recursion in the VPF algorithm. One might insist that each phrase $P\{l\}$ of length l has the form $P\{l\text{-}1\}w$, where w is one word and $P\{l\text{-}1\}$ is a phrase of length l-1. Since the phrase $P\{ l\text{-}1\}w$ is defined by the correlation between $P\{l\text{-}1\}$ and w, it is possible that the correlation exists between a non-phrase $P\{l\text{-}1\}$ and a word w. That is, $P\{l\text{-}1\}$ is not a phrase, but can be extended into a phrase of length l. If this is possible, it is also possible that even if there exists a phrase, $P\{l\}$, in a document, the phrase $P\{l\}$ could not be generated because $P\{l\text{-}1\}$ does not exist. For example, there exists a phrase "wireless powerful computer" in a web page. But, since "wireless powerful" is not a phrase, it is possible that the phrase could not be generated. However, if the phrase is meaningful/important enough, the sub phrases "wireless powerful" and "powerful computer" will be generated. Next, "computer" will be added to "wireless powerful". To relieve this problem, we calculate the threshold once at the beginning – this means the threshold is consistent. If the correlation value of "wireless powerful" is lower than the

value of "wireless powerful computer" then the shorter phrase will be removed at a pruning stage.

Our algorithm receives a sequence of words as input and returns meaningful phrases. It combines words into phrases until it generates no more phrases. Figure 12 illustrates the pseudo code for the variable-length phrase-finding algorithm (VPF). It uses three main variables – *List*, *SEQ*, and *Corr*. The *List* variable stores all collected phrases in a *Hash* attribute. Each element (phrase) of the *Hash* attribute keeps correlation value in *sim* attribute and the position list in *posi* attribute. VPF first makes a linked list (*SEQ*) with an input example. Each word and word position are stored in each node in the linked list. Then, all 1-gram distinct words are stored in *List*[1].*Hash*. *Corr* is a chosen correlation function. The *List* , *SEQ*, and *Corr* are passed to *BeSpecific* procedure and this finds all phrases. Once the phrases are acquired, they are pruned. The pruning process simply removes all sub-phrases that have a *sim* value lower than the *sim* value of the super-phrases.

The *BeSpecific* procedure receives five parameters: *List* that stores all phrases, *l* which is the length of phrases (initial value is 2), *thre* which keeps the calculated threshold value differentiating "strong" relations from "weak" relations (initialized to 0); *SEQ* which stores the linked list, and *Corr*. *BeSpecific* recursively creates new sequences by removing nodes that are not in the *Hash* table generated in the previous stage, and also by removing consecutive nodes whose lengths are shorter than *l*. Since it removes words that are not in the *Hash* table generated in the previous stage, there can be gaps between nodes. Once the new sequence is generated, it collects all *l*-grams having no gaps from the sequence. The threshold is calculated when *l*=2 only once by averaging all correlation values between the

first and second word in each element in *List*[2].*Hash*. The *for* loop removes any element with low correlation values (*sim*) from *Hash*. The *sim* property of *p* keeps the correlation between a sub phrase, *p*[1 .. *l*-1], and the last word, *p*[*l*], where *p* is a phrase consisting of *l* words. For example, if *p*="computer science seminar", then *p*[1 .. *l*-1]="computer science" and *p*[*l*]="seminar". The *Intersection* counts all adjacent points based on the distance. The intersection between the positions of *p*[1 .. *l*-1] and *p*[*l*] becomes the positions of *p*. If the intersection of *p* becomes 0, then the *p* is removed. The *BeSpecific* procedure recursively increases the phrase length *L* until *Hash* become empty. We can also apply various correlation functions in the place of *Corr*.

In order to improve the phrase-selection accuracy, we need to calculate for each word the percentage that a word can come before any other words and the percentage that the word can come after any other words, called pre-percentage and post-percentage respectively. The idea is that if a word occurred at the end of a sequence, then this word loses his one chance to come before any other words, so we adjust the pre-percentage of the word by deducting one from the number of occurrences of the word. The post-percentage is vice versa. We can view a string *S*[1..n] of *n* consecutive words as two sub strings, S_{pre}=*S*[1..*n*-1] and S_{post}=*S*[2..*n*]. Pre- and post-percentage of *w* can be computed in time O(1), when we know all the positions where *w* occurred:

$w_{pre\text{-}percentage}$= Frequency of *w* in S_{pre} / $|S_{pre}|$

$w_{post\text{-}percentage}$= Frequency of *w* in S_{post} / $|S_{post}|$.

```
Input: Example– a sequence of words
Output: Collected phrases
Function VPF (Example) return phrases
 SEQ– linked list, each node has a word and a position.
 List– array of Hash table, each word in a Hash has sim and posi
    attributes.
 Corr- a correlation function
1.  SEQ←linked list made by the input Example
2.  List[1].Hash←store all 1-grams, each word keeps its positions
3.  BeSpecific(List, 2, 0, SEQ, Corr)
4.  Prune phrases in the List
5.  Return all phrases in the List
End Function

Input: List- store array of Hash table
 L- length of phrase, initialized to 2
 thre- threshold value, initialized to 0
 SEQ- linked list, the size reduces every iteration
 Corr- correlation function
BeSpecific(List, L, thre, SEQ, Corr)
1.  if List[L-1].Hash is empty then stop
2.  SEQ←remove nodes that are not in the List[L-1].Hash and also remove
       consecutive nodes which length is shorter than L
3.  List[L].Hash←Collect all L-grams from SEQ
4.  if L=2 then thre←Average correlation across all words in List[2].Hash
5.  for each p in List[L].Hash do
6.    A ← pre-percentage of p[1..L-1]
7.    B ← post-percentage of p[L]
8.    p.sim←Corr(A,B,A∩B)
9.    remove any p for which p.sim is lower then thre
10.   p.posi←Intersection of p[1..L-1].posi and p[n].posi with distance of
       L-2
11.   remove any p for which p.posi=0
12. BeSpecific(List, L+1, thre, SEQ, Corr)
End Procedure
```

Figure 12. VPF algorithm

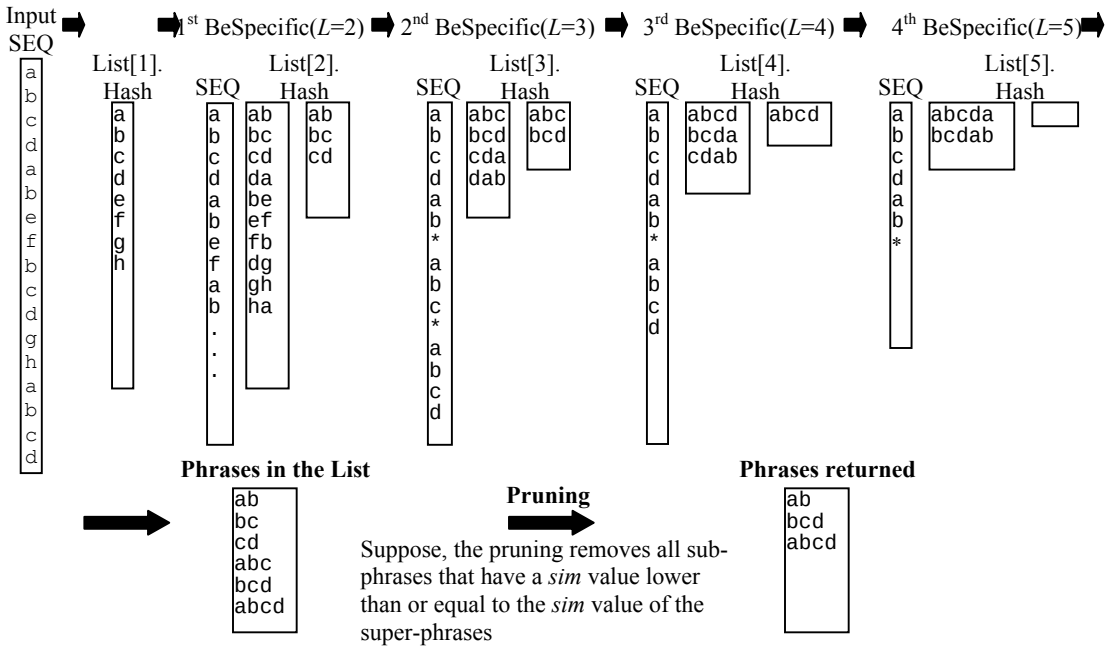


Figure 13. Example of VPF

The most expensive part in the *BeSpecific* procedure will be scanning a sequence when *L* is 2. Even though the *BeSpecific* is called *L* times, as *L* increases the size of the sequence decreases drastically. The *Corr* function has O(*T*) time complexity, where *T* is the position length. But, as T increases the number of element in a Hash decrease. We, therefore, can claim that the time complexity of VPF in general case is roughly: O(*S*), where *S* is the sequence size.

For example, in Figure 2 we have an input string *S* = "abcdabefbcdghabcd," in which each letter represents a word. The correlation function simply returns the frequency of a phrase using a threshold value of 1.5. *List*[1].*Hash* contains all distinct words. *List*[2].*Hash* originally contains all possible 2-word phrases, then removes any phrase that occur less than 2 times, resulting in {"ab", "bc", "cd"}. We remove all words that are not in the *List*[2].*Hash* from the 1st *SEQ* resulting in 2nd *SEQ*. *List*[3].*Hash* contains all 3-word phrases, and then remove "cda" and "dab" because they occur only once. When we came to *BeSpecific* for the 3rd time, we removed "*abc*" from the 3rd *SEQ*, because their consecutive length is less than 4 – the size of L increases every time we come to *BeSpecific*. When we run the 4th *BeSpecific*, we can remove all phrases ("abcda" and "bcdab") in *List*[5].*Hash*, because they occur only once. Since the 5th *Hash* is empty, the *BeSpecific* stops. After the *BeSpecific*, the *List* keeps {"ab", "bc", "cd", "abc", "bcd", "abcd"}. Suppose the pruning removes all sub-phrases that have a *sim* value lower than or equal to the *sim* value of the super-phrases. The occurrences of phrases are: "ab"-3, "bc"-3, "cd"-3, "abc"-2, "bcd"-3, and "abcd"-2. We remove "bc" and "cd" because "abc" subsumes them and has equal frequency. "abc" is removed by "abcd" with the same reason. The final returned phrases are {"ab", "bcd", "abcd"}.

3.1.2. Correlation Functions

The VPF algorithms build phrases; and correlation functions actually calculate the weight of a phrase. The correlation functions are important in terms of selecting more meaningful phrases. The VPF is able to cooperate with many different existing correlation functions, and it can be hard to choose one correlation function out of many. In this section, we describe several key properties of a good correlation function. Much of the statistical work in building multi-word features focuses on co-occurrence (Chen and Sycara, 1998; Rosenfeld, 1994). All correlation measures are not equally good at capturing the dependencies between different events (Tan et al., 2002). It is because each correlation function biases toward different individual event probabilities and joint probabilities. Piatetsky-Shapiro (1991) has proposed three key properties that a good correlation function, F, for events A and B should satisfy:

P1: if A and B are statistically independent, then F is 0;

P2: F monotonically increases with $P(A,B)$ when $P(A)$ and $P(B)$ remain the same;

P3: if $P(A)$ (or $P(B)$) increases when the rest of the parameters ($P(A,B)$ and $P(B)$ (or $P(A)$)) remain unchanged, then F monotonically decreases.

Statistical independence can be measured by the determinant operator, where $Det(A,B) = A \cap B \times A' \cap B' - A \cap B' \times A' \cap B$. Thus, a singular diagram is independent when its determinant is equal to zero (Tan and Kumar, 2000). Another important operation in finding phrases is distinguishing between positive and negative correlations ($P4$). Measuring their *cross*

product ratio (*CPR*) can assess the significance of the correlation between *A* and *B* (Rosenfeld, 1994) and is defined as:

$$\log CRP(A,B) = \log \frac{P(A,B)P(A',B')}{P(A',B)P(A,B')}$$

Negative correlation has a negative *log* CPR value. *P*4 is that *F* can distinguish positive and negative correlation of *A* and *B*. Since positive correlation is much more important than negative correlation in finding phrases, we only measured the change of correlation values over positive correlation.

Tan et al. (2002) illustrated those properties and extended them to each of the existing measures to determine if the existing measure satisfies the properties required (Kamber and Shinghal, 1996). Some of these properties have been extensively investigated in the data mining literature (Hilderman and Hamilton, 2001; Tan and Kumar, 2000; Tan et al., 2002). We examined 32 correlation functions of properties and cooperated them with phrase-finding algorithms. A complete list of the correlation functions to be examined in this study is given in Appendix 1.

3.2. Experiments

3.2.1. Experimental Data and Procedures

We use five New York Times articles and five web pages collected from our department server. We use web pages to test, because contents in a web page differ from the content found in normal article. The data used in this study is accessible at http://cs.fit.edu/~hkim/dissertation/dissertation.htm. The article size was about 2 pages and we asked 10 human subjects, other than the authors, to read the 10 articles. Each article contains about 1,300 words - 720 words after removing stop-words. Since we assigned 6 as a threshold of maximum phrases length for Ahonen's and Chan's, the total possible number of phrases for each article is approximately 3600 (=720×5). Of the 10 subjects, 4 were graduate students from a department of computer sciences and 6 were undergraduates with various majors.

We asked the 10 human subjects to choose their top 10 meaningful phrases for each article or web page. One might insist that the results will be different depending on how 10 humans are chosen. If all volunteers have the same background the matching rate will be higher than the normal case. However, since we are not comparing the algorithm with humans, but comparing among algorithms, it does not matter how we chose the 10 volunteers. Furthermore, since the algorithm finds phrases statistically that cover general human meaningfulness, we choose 10 human subjects arbitrarily.

The instruction that we gave them were:

```
Identify the top 10 "meaningful/important" phrases for each article.
Phrases are defined as two or more adjacent words that are meaningful,
for example, "computer science," "florida institute of technology," ...
The definition of meaningful is up to you.
```

We will measure the number of matches between the human subjects' selections and different correlation functions' selections as well as different phrase-finding algorithms.

We also count average matching of humans – in this case, we divided the sum by 9. There are cases for the human or algorithm to select less than 10 phrases. In order to be fair in these cases, we use an additional adjustment function. We also attempt to prevent a measure from being scored 1 by finding one phrase and having one matched phrase by chance – the results are too sensitive. We, therefore, decided to give a lower incentive as a measure finds fewer phrases 20% at the most. For example, if there are 5 matches out of 10, the number of matching is 5×1/10. If there are 5 matches out of 9, then we assigned 5×1/9. But, if there are 5 matches out of 8, then we assigned 5×1/8.5. The generalized formula is:

$$\text{Adjustment function (m, f)} = \frac{m}{8 + \frac{2}{2^{10-f}}}$$

, where m is the number of matched words and f is the number of selected words.

We also applied different correlation functions to Ahonen's algorithm to see if the difference of the performance depended on the correlation functions. Ahonen used two different correlation functions: conditional probability (Confidence, F11) for filtering phrases and mutual confidence (F32) for ordering the collected phrases determining which

phrase is more important than the other. Since he used fixed user-defined threshold (0.2) for filtering the phrases, we only varied the correlation function used for ordering phrases.

3.2.2. Evaluation Criteria

We evaluate the meaningfulness of phrases. We believe the closer a match comes to our set of human-selected phrases, the better the phrase-finding algorithm is in terms of finding meaningful phrases. To evaluate the correlation functions for each phrase-finding algorithm, we have two evaluation criteria: the number of *exact matches* and the number of *simple matches*. We have 128 methods (4 algorithms × 32 correlation functions) – the 4 algorithms are VPF, Chan's, Ahonen's, and Ahonen's with gap, and the correlation functions F1 through F32 are in Appendix A.

Table 10. With-pruning vs. without-pruning

Rank	Method	Avg. across humans and articles		Ratio of Improv.
		With-pruning	Without-pruning	
1	VPF_F25	**0.933**	0.883	5.7%
2	VPF_F16	**0.920**	0.866	6.2%
3	VPF_F28	**0.912**	0.871	4.7%
4	VPF_F8	**0.824**	0.707	16.5%
5	VPF_F10	0.819	**0.825**	-0.7%
6	VPF_F29	**0.814**	0.810	0.5%
7	VPF_F27	**0.812**	0.796	2.0%
8	VPF_F24	**0.812**	0.758	7.1%
9	VPF_F13	**0.772**	0.666	15.9%
10	VPF_F26	**0.726**	0.683	6.3%

The number of *exact matches* of a method is measured by the percentage of the matches between the human's and a method's. We count each match with a human's and

then average the 10 compared results. This counting approach assigns more weights to the more meaningful words – more meaningful word means that they were selected by more human subjects. If a phrase is selected by several human subjects, every match is counted. Therefore, finding more popular phrases increases the matching average. The number of matches will be very low, because only 10 phrases selected by a method and a human respectively are going to be compared.

The number of *simple matches* counts the matched phrases against the list collected by all human (i.e., the union of the words from the 10 human subjects). The list will be less then 100 because some phrases can overlap. *Simple match* is not directly related to finding more meaningful phrases, because this count removes the popularity information. We count this only to support the result of the *exact match*.

Comparison of the results is done using a matched-pair design (Robertson, 1981). In this design, the top ten phrases in the ranking generated are compared. The comparison, which simply identifies if one group of ten phrases is better than the other, is on the basis of precision in other words the number of matched phrases. This type of evaluation has the following advantages: It is realistic in the sense that many information retrieval systems are interested in the top group. Traditional recall/precision tables are very sensitive to the initial rank positions and evaluate entire rankings (Crotf and Das, 1989). Another advantage is that significance measures can be readily used.

3.3. Results and Analysis

Before comparing our algorithm with existing methods we need to decide whether to use pruning or not. After that we will be able to perform the comparison. In evaluating our method against related algorithms we use different scoring methods: exact match and simple match.

3.3.1. With-pruning vs. Without-pruning

The VPF algorithm has a pruning function. The results differed whether we used the pruning function or not. We compared them by comparing the top 10 best methods with *exact match* (Sec 6.2). By composing the algorithm and 32 correlation functions (in Appendix A), we generated 32 methods. We ranked the top 10 methods using "with pruning" and presented the corresponding results of "without-pruning" next in Table 10. The values are the average of matches across 10 human subjects and 10 articles. Most methods yielded improved results when they had been pruned. The top method VPF with F25 had improved its performance by 5.7%. With pruning is statistically significantly better than without-pruning with a 95% confidence interval (P=0.004).

3.3.2. Analysis with Exact Match

Because "with-pruning" achieves a higher matching rate than "without-pruning", we decided to use pruning in our algorithms for the rest of our experiment.

3.3.2.1. Top 10 Methods

The main purpose of the analysis in this section is to choose the best method. Which method is the best is the most interesting question. We averaged the results from 10 articles and 10 human subjects and sorted by the average to rank all 128 methods. We presented the results in Table 11 and included the rank, methods used, and the average. Each method was composed of an algorithm and a correlation function. Notice that, we also presented the results of previous methods. Ahonen used correlation function F32. He also introduced a method with gaps. The row Ahonen_gap represented the results using Ahonen's method allowing gaps within a phrase.

The best method was the combination of VPF and correlation functions F25 followed by F16 and F28 – all those three correlation functions satisfied Piatetsky-Shapiro's three desirable properties and distinguish positive from negative correlations. The best method VPF with F25 matched 0.93 phrases on average with the phrases selected by a human subject. In the next section we measured the average number of matching phrases between human subjects and compared those results to the results from methods.

Interestingly, VPF won the top 3. Chan's algorithm occupied the next ranks. Another observation was that the correlation functions F25, F16, and F28 that marked high rank with VPF also marked high rank with Chan's. This observation implied that the performance also depends on the correlation functions.

Table 11. Ranked by average across humans and articles – Exact match

Rank	Method	Avg.	Rank	Method	Avg.
1	VPF_F25	0.933	**13**	Ahonen_F6	0.797
2	VPF_F16	0.920	**15**	Ahonen_F10	0.779
3	VPF_F28	0.912	**15**	Ahonen_F11	0.779
4	Chans_F16	0.858	**15**	Ahonen_F12	0.779
5	Chans_F25	0.856	**15**	Ahonen_F17	0.779
6	Chans_F29	0.850	**15**	Ahonen_F26	0.779
7	Chans_F28	0.848	**20**	Ahonen_F20	0.774
8	VPF_F8	0.824	**20**	Ahonen_F23	0.774
9	VPF_F10	0.819	**24**	Ahonen_F32	0.767
10	VPF_F29	0.814	**105**	Ahonen_gap_F3	0.452

Table 12. Exact match across humans

	Avg. across 10 articles
Human best	1.48
Human avg.	1.30
Human worst	1.03

Ahonen's algorithm ranked 24 and Ahonen_gap 105. These methods matched 0.767 and 0.452 numbers of phrases with human subjects respectively. The low performance with gap is the same phenomenon as shown in (Ahonen et al., 1998). We conducted t-Test (paired two sample for means) between VPF with F25 and Ahonen with F32. There was a clear statistically significant difference between the two methods with 95% confidence (P=0.016). Therefore, we can conclude that VPF with F25 found statistically significantly more meaningful phrases than Ahonen's previous algorithm.

Ahonen's algorithm with other correlation functions received higher ranks such as F6, F10, F11, F12, F17, F26, and F20 as shown in Table 11. They all ranked 13, 15, and 20, which are higher than Ahonen's original method (24). This indicates Ahonen's algorithm can be improved upon by using different correlation functions.

3.3.2.2. Comparing with Human Subjects

To see the average number of matches among human subjects is interesting and also provides insight into interpreting the average number of matching by the algorithm. For instance, if an algorithm matches 1 on average and the human matches 7, then the performance of the algorithm is almost negligible no matter how much higher its performance is compared to others.

We presented the best, average, and worst matching human results in Table 3. The results told us that only 1.3 phrases out of 10 picked by a human subject matched with the phrases picked by the others on average. This is not an unrealistic result. Considering that each document has approximately 1,300 words, more than 7779 possible phrase combinations exist and each person has a different background, matching 1.3 phrases out of 10 on average is a reasonable result. Our method achieved a result (0.93), which was close to the typical human result. We also conducted a t-Test with the human average and VPF with F25. The human subjects' average was statistically significantly better than the best result obtained by the algorithm with a 95% confidence interval (P=0.02). It would be interesting to see if the worst case of human matching was higher than the algorithm's. The answer was no. It was not statistically significantly better than the machine's. This result

indicates that human matching is better than the matching of algorithms in general but not always.

Table 13. Ranked by average across humans and articles – Simple match

Rank	Method	Avg.	Rank	Method	Avg.
1	vpf_F28	3.696	**12**	ahonen_F10	3.195
2	vpf_F25	3.689	**12**	ahonen_F11	3.195
3	vpf_F13	3.672	**12**	ahonen_F12	3.195
4	vpf_F8	3.656	**12**	ahonen_F17	3.195
5	vpf_F27	3.575	**12**	ahonen_F26	3.195
5	vpf_F24	3.575	**17**	ahonen_F6	3.181
7	vpf_F21	3.377	**23**	ahonen_F2	3.025
8	vpf_F29	3.342	**24**	ahonen_F20	3.018
9	vpf_F16	3.321	**24**	ahonen_F22	3.018
10	chans_F29	3.282	**24**	ahonen_F23	3.018
22	chans_F25	3.053	**33**	ahonen_F32	2.934
			118	ahonen_gap_F32	1.755

Table 14. Simple match across humans

	Avg. across 10 articles
Human best	6.3
Human avg.	5.6
Human worst	4.7

3.3.3. Analysis with Simple Match

This *simple match* count showed similar ranking to the *exact match*. VPF with F28 followed by F25 and F13 had the top matching rates: 3.70, 3.69, and 3.67 respectively as shown in Table 13. Since simple match uses a list of meaningful phrases by taking the

union of phrases selected by the 10 human subjects, average number of matching phrases is higher than the average by *exact match*. Chan's with F25 ranked 22 (3.05 matching rate), Ahonen without gap ranked 33 (2.93), and Ahonen with gap ranked 118 (1.76) out of 128. Chan's original method ranked 22 (3.053). These results also told us VPF found more phrases than Ahonen's and Chan's. The result from *simple match* also indicated that Ahonen's algorithm could be improved by incorporating different correlation functions.

We also attempted to compare the methods' results with the results from humans. Human matched the list 5.6 out of 10 on average; the best and worst cases are 6.3 and 4.7 as shown in Table 14. The result 3.69 of method VPF with F25, which was the highest score with exact match, was quite significant considering that we only used the statistical information.

3.4. Summary

We proposed a variable-length phrase-finding algorithm, which find more meaningful phrases – VPF – than older methods – Ahonen's and Chan's algorithms. We also coordinated these algorithms with 32 different correlation functions. They regenerate sequences recursively with the words selected in the previous stage and search for increased length of phrases in time $O(n_w)$, where n_w is the number of distinct words in a page. Since our algorithm uses average as a threshold and stops when the length of phrases does not increase, no user-defined parameter is required.

In order to choose the best method, we conducted an experiment by asking 10 human subjects to select 10 phrases from 10 different articles. We compared the number of matching phrases chosen by a method to those phrases chosen by 10 human subjects. By

comparing the top 10 best measures (matched-pair design (Robertson, 1981)), we observed that when we add pruning, the algorithm (VPF) had improved performance.

We concluded that VPF with F25 found a statistically significantly greater number of meaningful phrases than Ahonen's previous method. We suspect the filtering stage of Ahonen's algorithm filtered many meaningful phrases out or their weighting scheme using the length of a phrase and tightness (Ahonen et al., 1998) distracted the correlation value of a phrase. Interestingly, the correlation functions F25 and F28 were both included in the top 10 in both *exact match* and *simple match*. This result indicates the correlation functions F25 and F28 had higher matching rates than the other correlation functions. These two correlation functions both satisfied desirable properties for phrases. We can also improve Ahonen's algorithm by incorporating correlation functions F10, F11, F12, F17, F26, F6, and F20. Those functions resulted in a higher match of average scores for both *exact match* and *simple match* experiments.

The performance of our method varied depending on the articles selected. We currently do not understand the reason for the variance in performance over different articles. We assume it is due to the intrinsic characteristics of an article, because the human subjects' results are also different depending on the articles. Phrases in VPF grow backwards; however, in the future we will devise an algorithm that grows both forwards and backwards.

Chapter 4

Analysis of Desirable Properties of Correlation Functions between Two Events

Before selecting a correlation function, it is important to check whether a correlation function satisfies basic desirable properties of a correlation function. Likewise knowing the characteristics of a correlation function is important. We propose two new desirable properties that a correlation function should satisfy. These new properties will help understand the general characteristics of correlation functions and help to choose or devise a right correlation function for an application domain. Correlation functions can be used for finding phrases, building profile, and devising scoring functions in our system as shown in Figure 14. We tested with 32 correlation functions to see which one satisfies what desirable properties.

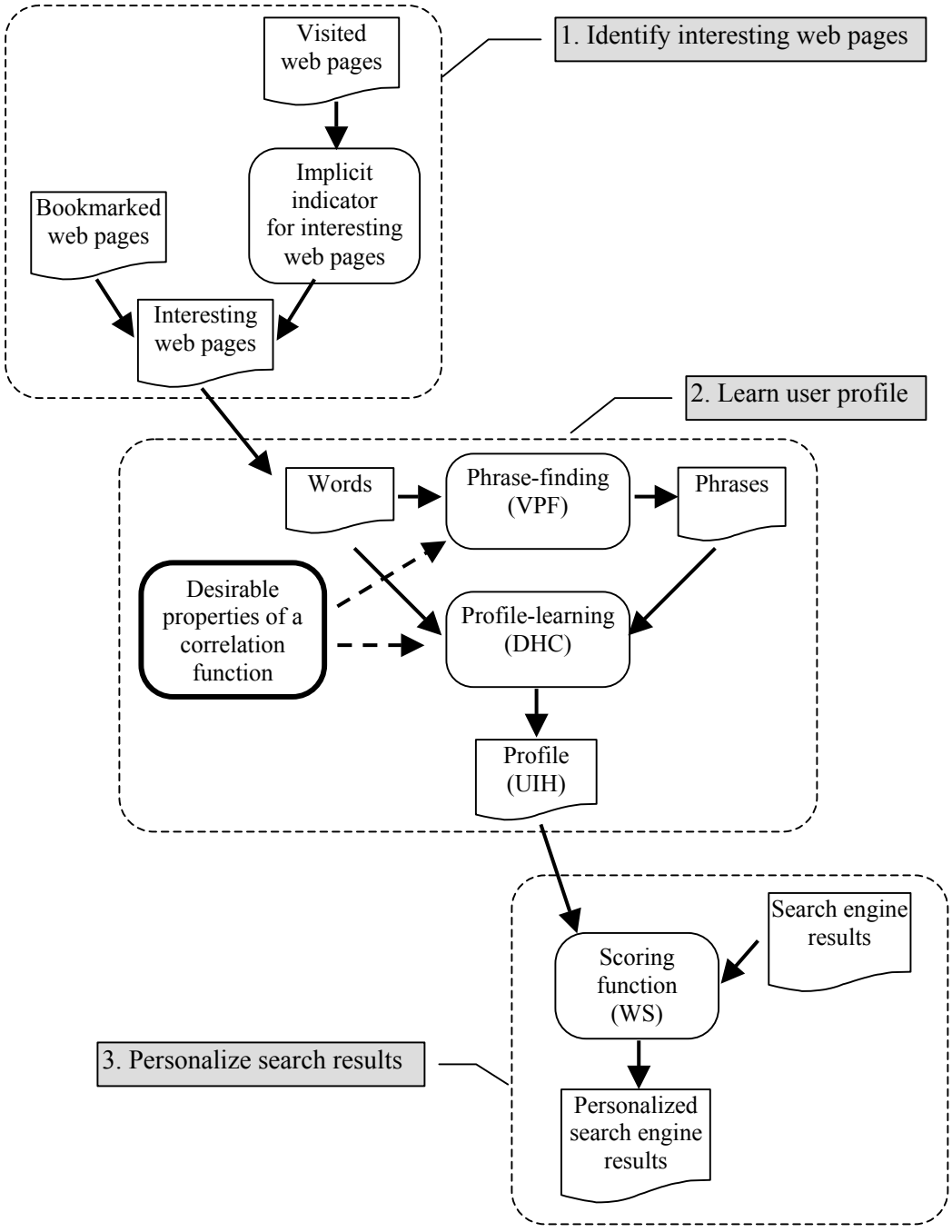


Figure 14. Desirable properties in the framework of web personalization

4.1. Desirable Properties of a Correlation Function

In this section, we describe several key properties of a correlation function. Correlation functions differ in their ability to capture the dependencies between variables, because each correlation function has its own bias in preferring a set of diagrams to another. The dependencies between variables can be described in a Venn diagram as shown in Figure 15 – A, B, $A \cap B$, and $A \cup B$.

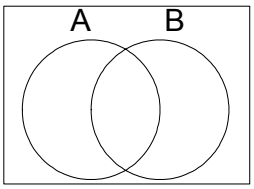


Figure 15. Venn diagram

Piatetsky-Shapiro (1991) proposed three key properties that a good correlation function, F, should satisfy:

P1: if A and B are statistically independent, then F is 0;
P2: F monotonically increases with $P(A,B)$ when $P(A)$ and $P(B)$ remain the same;
P3: if $P(A)$ (or $P(B)$) increases when the rest of the parameters, $P(A,B)$ and $P(B)$ (or $P(A)$), remain unchanged, then F monotonically decreases.

Tan et al. (2002) illustrated those three properties and the extent to which each of the existing measure satisfies the properties. Some of these properties have been extensively investigated in the data mining literature (Hilderman and Hamilton, 2001; Tan and Kumar, 2000).

4.1.1. Enhancing Property 1

The detection of statistical independence is a desirable property – Piatetsky-Shapiro (1991) specified this as his first desirable property (P1). However, should the correlation function return only 0?

When two events, A and B, are statistically independent, some functions return 0, but some functions return a different value such as 1. Each column in Table 19 is for each correlation function and we used "o" mark as satisfaction of P1 in the P1 row. The "value" row signifies the return value. For instance, the test functions *Odds*, *Conv*, *Inte*, and *Coll* in Appendix 1 return 1 when two events are statistically independent as shown in Table 19. Therefore, we argue that the definition of P1 is too strict. We revise the definition of P1 to be:

P1: F can distinguish statistical independence of A and B.

The revised definition of P1 is used for the duration of this Chapter.

4.1.2. Additional Desirable Properties

We propose and investigate additional properties desirable for a correlation function. When the universe is fixed, three events – A, B, and $A \cap B$ – can enumerate all possible cases. We list seven possible cases in Table 15. The notation "↑" indicates increase, "↓"indicates decrease, and "-"indicates no change in the value of a correlation function. We add two other possible cases – two variables increase at the same ratio (P4 and P5 in Table 15).

Case 1 has variable A fixed, B fixed, and $A \cap B$ increasing. Piatetsky-Shapiro (1991) refers to this case as P2, so we follow his designation. We show the proper change of a correlation function in the "Proper change" column. Other cases can be interpreted in the same way. The proper change of case 1 is increasing (Piatetsky-Shapiro, 1991). Case 2 and 3 have the same property name (P3) by similarity – variable A and B could be alternated with each other. The proper change of P3 is decreasing (Piatetsky-Shapiro, 1991).

Table 15. All possible cases (with the same incremental ratio)

Case	A	B	$A \cap B$	Property name	Proper change
1	–	–	↑	P2	↑
2	↑	–	–	P3	↓
3	–	↑	–		
4	↑	↑	–	P4	↓
5	↑	–	↑	P5	↑
6	–	↑	↑		
7	↑	↑	↑		

Case 4 appears to be the same as the inverse of P2. But not all correlation functions return the same results (e.g., *Supp* in Appendix 1 increases for case 1 but no change for case 4). The value of a correlation function should decrease as the value of $A \cup B$ increases when $A \cap B$ remains unchanged in the formula of $\frac{A \cap B}{A \cup B}$. As A and B increase, the value of $\frac{A \cap B}{A \cup B}$ decreases when $A \cup B = A + B - A \cap B$. Therefore, property 4 (P4) states that if $P(A,B)$ increases when $P(A)$ and $P(B)$ remain unchanged, then F monotonically decreases.

An example of case 4 is shown in Table 16. The union is 100. Variable A and B increase with the same incremental ratio; $A \cap B$ remains unchanged. Intuition suggests as A and B increase, the value of a correlation function is expected to decrease.

Table 16. An example for property 4

A	B	$A \cap B$	Incremental ratio
20.0	30.0	10.0	1.0
22.0	33.0	10.0	1.1
24.0	36.0	10.0	1.2
30.0	45.0	10.0	1.5
34.0	51.0	10.0	1.7
40.0	60.0	10.0	2.0

Case 5 and 6 have the same property name (P5) due to their similarity. The proper change in a correlation value for P5 is monotonically increasing. In the formula of $\frac{A \cap B}{A + B - A \cap B}$, where A and B are greater than $A \cap B$ and one of A and B are fixed, as B (or A) and $A \cap B$ increase together at the same ratio, the whole value should increase. The P5 states that if $P(A)$ (or $P(B)$) and $P(A,B)$ increase at the same ratio, then F monotonically increases.

We illustrate cases 5 and 6 with an example below in Table 17. Since both cases are the same, we will only use case 5. Cases can have two possible sub-scenarios: A is smaller than B, or A is bigger than B. The union is 100. Variable A and $A \cap B$ increase with the same incremental ratio; B remains unchanged. In the case of $A<B$, the value of correlation function is expected to increase, because the ratio between B and $A \cap B$ increases, while the ratio between A and $A \cap B$ remains the same. In the case of $A>B$, A seemingly subsumes B gradually, because B remains the same while $A \cap B$ grows.

Consider the cases where A_1=5.0, A_2=10.0, B=10.0, $A_1 \cap B$=3.0, $A_2 \cap B$=6.0 as in Table 17. The correlations of the two cases ($F(A_1,B,A_1 \cap B)$ and $F(A_2,B,A_2 \cap B)$) may have different values. The *log CPR* (Rosenfeld, 1994) values for the two cases are 4.2 and 5.0 respectively, which assess the significance of the correlation between A and B. However, for example *Inte* in Appendix 1 satisfies all previous 3 desirable properties (P1 – P3); but, it returns the same value, 6, for both cases (case of $(A_1,B,A_1 \cap B)$ and case of $(A_2,B,A_2 \cap B)$). Our P5 is to verify whether a correlation function can tell the difference between the two cases. This determines that the P5 is critical for measuring the characteristics of correlation functions.

Table 17. An example for property 5

Case of $A<B$			**Case of $A>B$**			**Incremental ratio**
A	***B***	***A∩B***	***A***	***B***	***A∩B***	
5.0	10.0	3.0	20.0	10.0	4.0	1.0
5.5	10.0	3.3	22.0	10.0	4.4	1.1
6.0	10.0	3.6	24.0	10.0	4.8	1.2
7.5	10.0	4.5	30.0	10.0	6.0	1.5
8.5	10.0	5.1	34.0	10.0	6.8	1.7
10.0	10.0	6.0	40.0	10.0	8.0	2.0

Case 7 is defined as the values of A, B, and $A \cap B$ increase monotonically together at the same ratio. It is difficult to define what is a desirable change of the correlation function for this case. This property is somewhat related to the frequency. For example, too frequent and too rare words are usually removed in some researches (Croft et al., 1991; Zamir and Etzioni, 1998). Due to the uncertainty inherent in case 7, we are unable to determine any proper changes to this property.

4.2. Experiments

4.2.1. Experimental Data and Procedures

If all those correlation functions that satisfy our new 2 desirable properties also satisfy the previous 3 desirable properties, our proposed properties would not be a substantial improvement in characterizing correlation functions. Therefore, we collected 32 correlation functions to see which correlation functions only satisfy the previous 3 desirable properties. A complete listing of the correlation functions examined for further properties in this Chapter was given in Appendix 1. In order to verify whether a correlation function satisfies desirable properties, we systematically generated 923 different test cases (combinations of A, B, and $A \cap B$). For P2 we generated 76 test cases, P3 had 399 test cases, P4 had 35 test cases, and P5 had 413 test cases. Generating test cases P3 and P5 were more complicated, because both had two different sub cases. We wrote our program with Excel macro and ran on Intel Pentium 4 CPU. The detailed test cases and source code are available on the web: http://cs.fit.edu/~hkim/dissertation/dissertation.htm.

We added *dMAX*, *AEMI3*, *dMIN*, and *dMIN2* correlation functions to an experiment. The *dMAX* and the *dMIN* were cooperating with the *Supp*. The *dMIN2* emphasized the original *MIN* value. The *AEMI3* function was similar to *AEMI4* except that *AEMI3* did not include the negative relation part, $A'B' \times \log(A'B'/A' \times B')$. For example, the two matrixes in Figure 16 had different value assignments – basically the positive and negative intersection values were exchanged. An *AEMI4* weighed two tables the same, even though Matrix A had a value of 5 for $A \cap B$ and Matrix B had a value of 75 for $A \cap B$. However, in some application domains the positive relation, $A \cap B \times \log(A \cap B/(A \times B))$ is

more important than the negative relation, which prefers Matrix B to Matrix A. Therefore, *AEMI3* removed the negative relation part from *AEMI4*.

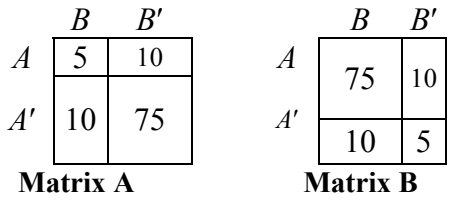


Figure 16. Contingency matrix

4.2.2. Evaluation Criteria

In addition to the proposed properties (P1-P5), another important property was the ability to distinguish between positive and negative correlations (P6). Tan et al. (2002) described this property in detail. Positive correlation is more important than negative correlation in some application domains. Therefore, we also checked correlation functions whether they do or do not satisfy P6.

Statistical independence (P1) can be measured by the determinant operator, $\mathrm{Det}(A,B) = A \cap B \times A' \cap B' - A \cap B' \times A' \cap B$. Thus, a singular Venn diagram is independent when whose determinant is equal to zero (Tan et al., 2002). Measuring their *cross product ratio* (*CPR*) can assess the significance of the correlation between A and B (Rosenfeld, 1994) and is defined as:

$$\log CPR(A,B) = \log \frac{P(A,B)P(A',B')}{P(A',B)P(A,B')}.$$

Negative correlation function would have a negative *log* CPR value. The definition of P6 was that F can distinguish between positive and negative correlations of A and B. We also

tested whether the result of each correlation function was normalized or not. P7 is that F returns normalized results. The total properties that we summarized are listed in Table 18. Normalized functions have return values within certain boundaries such as between 0 and 1.

Table 18. Summary of correlation functions' properties

Property	Description
P1	F can distinguish statistical independence of A and B.
P2	F monotonically increases with $P(A,B)$ when $P(A)$ and $P(B)$ remain the same
P3	If $P(A)$ (or $P(B)$) increases when the rest of the parameters, $P(A,B)$ and $P(B)$ (or $P(A)$), remain unchanged, then F monotonically decreases.
P4	If $P(A,B)$ increases when $P(A)$ and $P(B)$ remain unchanged, then F monotonically decreases.
P5	If $P(A)$ (or $P(B)$) and $P(A,B)$ increase at the same ratio, then F monotonically increases.
P6	F can distinguish positive and negative correlations of A and B.
P7	F returns normalized results.

We measured mainly whether a correlation function shows consistent increasing pattern or decreasing pattern under a condition for each property. The changes with negative relations are disregarded. Since we can detect positive and negative correlations it would be more practical to check over positive correlations.

4.3. Results and Analysis

The purpose of this experiment is to test whether those correlation functions that satisfy our new 2 desirable (new properties) properties also satisfy 3 established desirable properties (old properties). If there are correlation functions that satisfy only one of either the previous 3 properties or our 2 new properties, then the experiment provides evidence that our new properties will be useful for distinguishing properties of correlation functions.

This analysis based upon 3 categories. First, we compared old properties and new properties based upon the number of correlation functions that satisfies those properties. Second, old properties and new properties were compared based upon the number of functions that satisfied the two categories of properties upon the satisfaction of P1. Third was regarding the satisfaction of P6. After that, we summarized whether correlation functions return normalized results.

4.3.1. Comparing Properties: Old verses New

We compared how many correlation functions satisfied old properties and how many satisfied new properties. Each property of a correlation function and the desirable changes in the function for the properties P2 though P5 were given in Table 18. Table 19 contained the detailed results of our experiment designed to test correlation functions and the five desirable properties of correlation functions. If a given function satisfies a particular property, it receives an "o" notation, and "×" if the function does not satisfy the property. Some properties could have 2 conditions ($A>B$ or $A<B$) depending on which variable was greater.

Twenty correlation functions – *Coef*, *Odds*, *YulQ*, *YulY*, *Kapp*, *Mutu*, *JMea*, *Gini*, *Conv*, *Inte*, *Piat*, *Certa*, *Added*, *Coll*, *Klos*, *MI*, *EMI*, *AEMI4*, *dMI*, and *AEMI3* – satisfied old properties. Nineteen correlation functions – *Coef*, *Odds*, *YulQ*, *YulY*, *Kapp*, *Mutu*, *JMea*, *Gini*, *Piat*, *Coll*, *Jacc*, *Klos*, *EMI*, *AEMI4*, *dMAX*, *dMI*, *AEMI3*, *dMIN*, and *MuConf* – satisfied new properties. Fifteen correlation functions – *Coef*, *Odds*, *YulQ*, *YulY*, *Kapp*, *Mutu*, *JMea*, *Gini*, *Piat*, *Coll*, *Klos*, *EMI*, *AEMI4*, *dMI*, and *AEMI3* – satisfied all five desirable properties.

We summarized the results in a Contingency matrix in Figure 17. The numbers outside the box were the sum of rows and columns. Out of 32 total correlation functions tested, 20 correlation functions satisfied old properties (P1-P3); 19 satisfied our new properties (P4, P5); 15 satisfied both desirable properties (P1-P5); 5 functions satisfied old properties only, and 4 functions satisfied new properties only. Since our properties could distinguish 5 correlation functions out of 20 functions that satisfied old properties (5/20=25%), it determined that our properties were independent from previous 3 desirable properties.

	Satisfying old properties	Unsatisfying old properties	
Satisfying new properties	15	4	19
Unsatisfying new properties	5	8	13
	20	12	32

Figure 17. Contingency matrix of desirable properties over all functions

4.3.2. Comparison Based upon Property 1

If a correlation function can detect the statistical independence of a diagram, it satisfies P1. The correlation functions that satisfy old properties and new properties were counted under the condition of satisfying P1. We created contingency matrix of the correlation functions that satisfied P1 in Figure 18. Twenty correlation functions satisfied P1. Interestingly, the correlation functions that satisfied P1 also satisfied P2 and P3. The correlation functions that satisfied P2 and P3 were dependent on the correlation functions that satisfied P1. This means P1 alone can explain P2 and P3. In other words, P1 can subsume P2 and P3, which reveals the nonessential nature of P2 and P3. When verifying correlation functions, P1 may be the only necessary reference property.

	Satisfying old properties	Unsatisfying old properties	
Satisfying new properties	15	0	15
Unsatisfying new properties	5	0	5
	20	0	20

Figure 18. Contingency matrix of desirable properties over property 1

4.3.3. Comparison Based upon Property 6

In Table 19, an “o” is placed in each column, for P6 if the correlation function can distinguish positive and negative relations. The correlation functions that satisfied old properties or new properties was counted and shown in the contingency matrix (Figure 19).

Of the 12 functions satisfying P6, 8 satisfied P1-P5; only 4 satisfied old properties (P1-P3) without satisfying new properties (P4 and P5).

All correlation functions satisfying P6 also satisfied old properties. It might be unnecessary to test a function for P1-P3 when P6 encompasses all three. Our properties (P4 and P5) tested an alternative aspect of the correlation functions, which is unrelated to P6. In the contingency matrix, we compared 12 correlation functions, which satisfied old properties to new properties (Table 19). Of the 12 functions, only 8 satisfied our new properties. Clearly our properties were testing an alternative aspect of correlation functions.

	Satisfying old properties	Unsatisfying old properties	
Satisfying New properties	8	0	8
Unsatisfying New properties	4	0	4
	12	0	12

Figure 19. Contingency matrix of desirable properties over property 6

4.3.4. Normalized Results – Property 7

Normalization helps users analyze or understand the data. The P7 is not a highly desirable property because normalization is unrealistic for many correlation functions. We included the results of P7 testing only to help those users who prefer normalized results.

A comparison of P7 to both old properties and new properties may prove illustrative. Correlation functions that produced normalized results were assigned an “o” in Table 19. Out of 32 correlation functions, 22 functions satisfied P7. Ten functions satisfied

both old properties and P7; fourteen functions satisfied both new properties and P7. The results also indicated that the characteristics of correlation functions were unrelated to function normality.

4.4. Summary

In order to select the right correlation functions for an application out of a set of well-known correlation functions, the characteristics of each function must be compared. Piatetsky-Shapiro (1991) opened new research areas by proposing three basic desirable properties for functions (P1-P3). Tan et al. (2002) compared correlation functions using the three established desirable properties and other existing function properties. Our research on two-variable Venn diagrams led us to identify two more desirable properties of correlation functions (P4 and P5). Property 4: If $P(A,B)$ increases when $P(A)$ and $P(B)$ remain unchanged, then F monotonically decreases. Property 5: If $P(A)$ (or $P(B)$) and $P(A,B)$ increase at the same ratio, then F monotonically increases.

Consider the cases where A_1=5.0, A_2=10.0, B=10.0, $A_1 \cap B$=3.0, and $A_2 \cap B$=6.0 (Table 17). The correlations of the two cases, $F(A_1,B,A_1 \cap B)$ and $F(A_2,B,A_2 \cap B)$, may have different values. The *log CPR* values, which assess the significance of the correlation between A and B, are 4.2 and 5.0 respectively (Rosenfeld, 1994). But a deficiency exists. For example, the function *Inte* satisfies P1-P3, but returns the same value of 6 in both cases. Our proposed property 5 can tell the difference between the two cases, and is critical for measuring the characteristics of correlation functions.

In addition, the definition of P1 used by Piatetsky-Shapiro (1991) lacks some descriptive power. For example, some correlation functions (e.g., *Odds*, *Conv*, *Inte*, and

Coll in Appendix 1) produced a score of 1, even when two events were statistically independent. We revised the definition of P1 to improve its descriptive power.

P1: F can distinguish statistical independence of A and B.

In order to see whether those correlation functions that satisfy our two new desirable properties also satisfy previous three desirable properties, we collected 32 correlation functions as shown in Appendix 1 and used systematically generated test cases for testing each property. The results indicated that our two new desirable properties were more descriptive than the previous three old desirable properties. It was because all correlation functions that satisfy P1 also satisfied P2 and P3, and all correlation functions that satisfied P6 also satisfied P1-P3; but our properties were independent from P1 and P6. We insist that these two properties are important in terms of understanding the correlations of two variables and characterizing an appropriate correlation function. The summarized results not only of P1-P5 but also of P6 that measured negative/positive correlation and the result of P7 that checked which correlation function returned a normalized return value will help reader compare characteristics of correlations functions.

Our experiment was limited to positive correlations for our web personalization since many applications depend on positive correlation. We will extend our analysis to negative correlation as well.

Table 19. Properties of correlation functions

Property	Conditions	1	2	3	4	5	6	7	8	9	10	11
		Coef	**Good**	**Odds**	**YulQ**	**YulY**	**Kapp**	**Mutu**	**JMea**	**Gini**	**Supp**	**ConMa**
P1		o	x	o	o	o	o	o	o	o	x	x
	Value	0		1	o	o	o	o	o	o		
P2	*A=B*	o	o	o	o	o	o	o	o	o	o	o
P3	*A<B*	o	x	o	o	o	o	o	o	o	x	x
	A>B	o	x	o	o	o	o	o	o	o	x	o
P4	*A=B*	o	o	o	o	o	o	o	o	o	x	o
P5	*A<B*	o	x	o	o	o	o	o	o	o	o	o
	A>B	o	x	o	o	o	o	o	o	o	o	x
P6		o	x	x	o	o	o	x	x	x	x	x
P7		x	o	x	o	o	o	o	o	o	o	o
	Scope		0-1						0-.5	0-.5	0-.5	0-1

Property	Conditions	12	13	14	15	16	17	18	19	20	21	22
		Lapl	**Conv**	**Inte**	**Cosi**	**Piat**	**Certa**	**Added**	**Coll**	**Jacc**	**Klos**	**MI**
P1		x	o	o	x	o	o	x	o	o	o	o
	Value		1	1		o	o	o	1		o	o
P2	*A=B*	o	o	o	o	o	o	o	o	o	o	o
P3	*A<B*	o	o	o	o	o	o	o	o	o	o	o
	A>B	x	o	o	o	o	o	o	o	o	o	o
P4	*A=B*	o	o	o	o	o	o	o	o	o	o	o
P5	*A<B*	x	x	x	x	o	x	x	o	o	o	x
	A>B	o	o	x	o	o	o	o	o	o	o	x
P6		x	x	x	x	o	o	x	o	o	x	x
P7		o	x	x	o	o	x	o	x	o	o	x
	Scope	0-1			0-1	0-1	0-1	0-1		0-1	0-1	

Property	Conditions	23	24	25	26	27	28	29	30	31	32	
		StcMi	**EMI**	**Aemi4**	**dMAX**	**dMI**	**AEMI3**	**dMIN**	**dMIN2**	**NegCos**	**MuCon**	
P1		x	o	o	x	o	o	x	o	o	x	
	Value		o	o		o	o					
P2	*A=B*	o	o	o	o	o	o	o	x	o	o	
P3	*A<B*	o	o	o	o	o	o	x	x	o	o	
	A>B	x	o	o	x	o	o	o	o	o	o	
P4	*A=B*	o	o	o	o	o	o	o	o	o	o	
P5	*A<B*	x	o	o	o	o	o	o	x	x	o	
	A>B	o	o	o	o	o	o	o	x	x	o	
P6		o	o	x	x	o	x	o	o	x	x	
P7		o	o	x	o	o	x	o	o	x	o	
	Scope	0-1	0-1		0-1	0-1		0-1	0-1		0-1	

Chapter 5

Personalized Ranking of Search Results with Implicitly Learned User Interest Hierarchies

Web search engines are usually designed to serve all users, without considering the interests of individual users. Personalized web search incorporates an individual user's interests when deciding relevant results to return. We propose to learn a user profile, called a user interest hierarchy (UIH), from web pages that are of interest to the user. The user's interest in web pages will be determined implicitly, without directly asking the user. Using the implicitly learned UIH, we study methods that (re)rank the results from a search engine. Experimental results indicate that our personalized ranking methods, when used with a popular search engine, can yield more relevant web pages for individual users. This process is depicted in Figure 20.

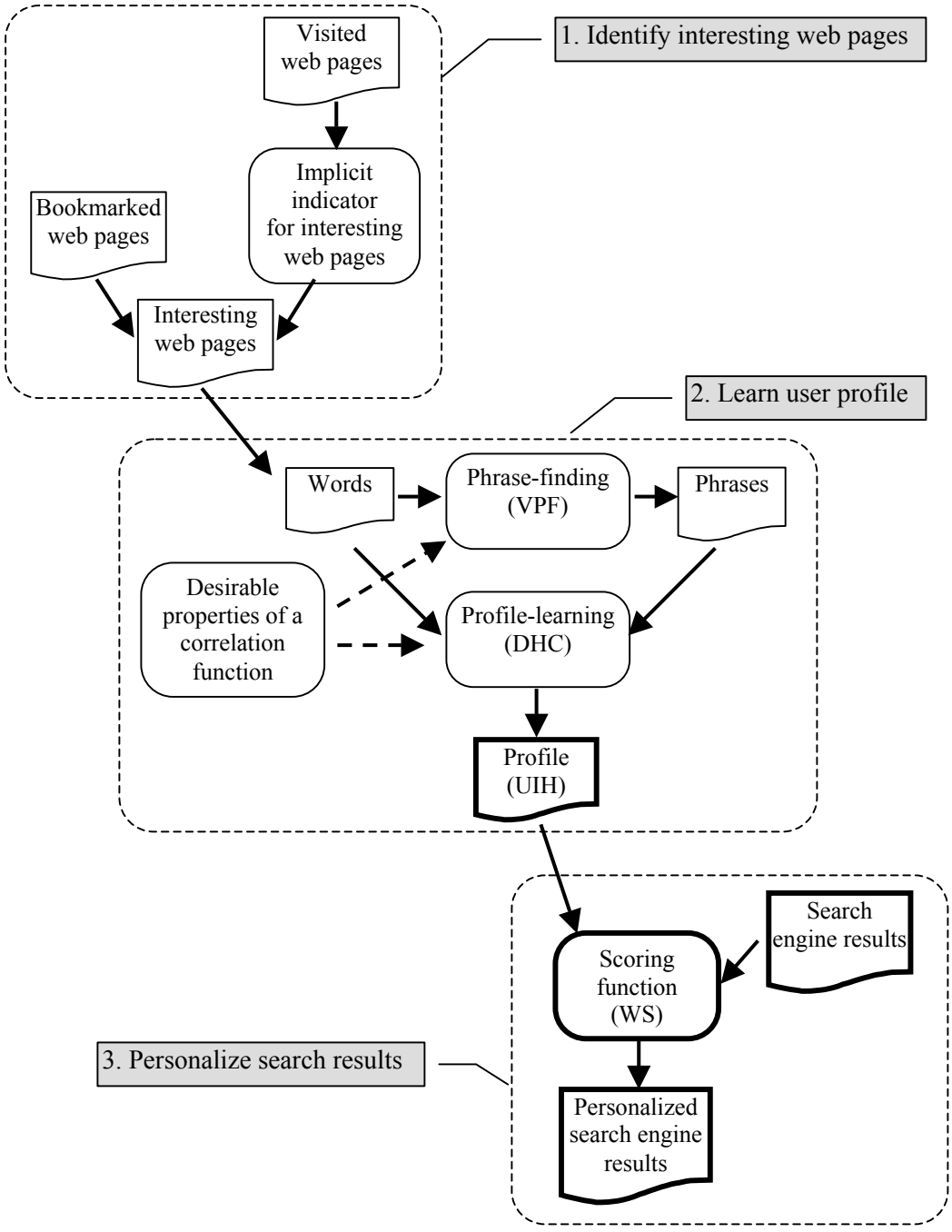


Figure 20. Devising a scoring function in the framework of web personalization

5.1. Personalized Results

Personalization of web search involves adjusting search results for each user based on his or her unique interests. Our approach orders the pages returned by a search engine depending on a user's interests. Instead of creating our own web search engine, we retrieved results from Google (Google, 2005). Since the purpose of this Chapter is to achieve a personalized ordering of search engine results, we can score a page based on the user profile and the results returned by a search engine as shown in the dashed box in Figure 21.

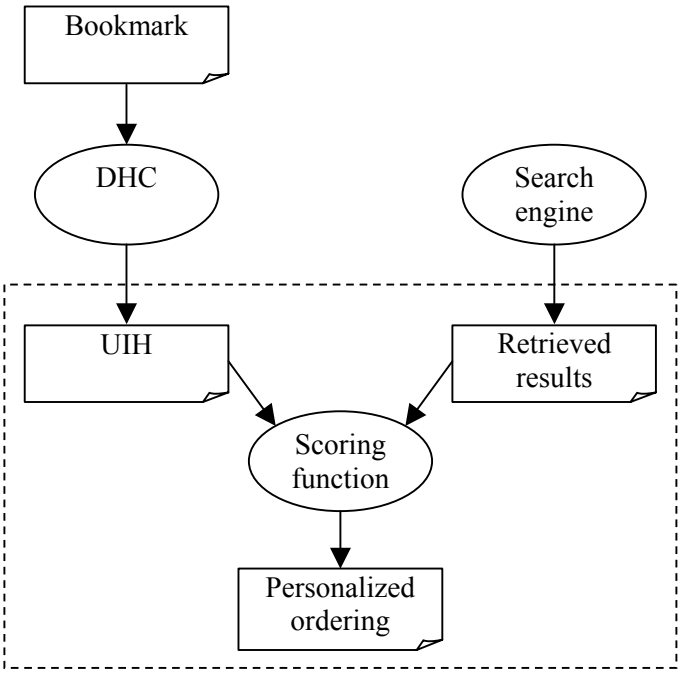


Figure 21. Diagram of Scoring

To build the user profile, called User Interest Hierarchy (UIH), we use the web pages in his/her bookmarks (Li et al., 1999; Maarek and Ben-Shaul, 1996) and the Divisive Hierarchy Clustering (DHC) algorithm (Kim and Chan, 2003). A UIH organizes a user's interests from general to specific. Near the root of a UIH, general interests are represented by larger clusters of terms while towards the leaves, more specific interests are represented by smaller clusters of terms. The root node contains all distinct terms in the bookmarked web page. The leaf nodes contain more specifically interesting terms. The relations between terms are calculated based on the co-occurrence in the same web page.

An example of a UIH is shown in Figure 3. Each node (cluster) contains a set of words. The root node contains all words that exist in a set of web pages. Each node can represent a conceptual relationship if those terms occur together at the same web page frequently, for example 'perceptron' and 'ann' (in italics) can be categorized as belonging to neural network algorithms, whereas 'id3' and 'c4.5' (in bold) in another node cannot. Words in these two nodes are mutually related to some other words such as 'machine' and 'learning'. This set of mutual words, 'machine' and 'learning', performs the role of connecting italicized and bold words in sibling nodes and forms the parent node. We illustrate this notion in the dashed box.

This Chapter focuses on devising a scoring method that receives two inputs (UIH and retrieved results) and one output (personalized ranking).

5.2. Approach

In order to provide personalized, reordered search results to a user, we need to score each page depending on personal interests. Therefore, the goal is to assign higher

scores to web pages that a user finds more interesting. This section explains how to score a retrieved web page using a user's UIH. First, we explain the basic characteristics for each matching term. Second, based on the characteristics, we propose functions to score a term. These functions determine how interesting a term is to a user. Third, based on the score and the number of the matching terms, we calculate an overall score for the page. Last, since the search engine provides a score/ranking for a web page, we incorporate this ranking into our final score of the web page.

5.2.1. Four Characteristics of a Matching Term

Given a web page and a UIH, we identify matching terms (words/phrases) that reside both in the web page and in the UIH. The number of matching terms is defined m, which is less than the number of total distinct terms in the web page, n_t, and the number of total distinct terms in the UIH, n_u.

Each matching term, t_i, is analyzed according to four characteristics: the level of a node where a term belongs to (D_i), the length of a term such as how many words are in the term (L_i), the frequency of a term (F_i), and the emphasis of a term (E_i). D and L can be calculated while building a UIH from the web pages in a user's bookmark. Different web page has different values for F and E characteristics. We estimate the probability of these four characteristics and based on these probabilities, we approximate the significance of each matching term.

5.2.1.1. Level/Depth of a UIH Node

A UIH represents general interests in large clusters of terms near the root of the UIH, while more specific interests are represented by smaller clusters of terms near the leaves. The root node contains all distinct terms and the leaf nodes contain small groups of terms that represent more specific interests. Therefore, terms in more specific interests are harder to match, and the level (depth) where the term matches indicates significance. For example, a document that contains terms in leaf nodes will be more related to the user's interests than a document that contains the terms in a root node only. If a term in a node also appears in several of its ancestors, we use the level (depth) closest to the leaves.

There is research that indicates user-defined query scores can be used effectively (Salton and Waldstein, 1978; Harper, 1980; Croft and Das, 1989). From the acquisition point of view, it is not clear how many levels of importance users can specify if we ask a user directly. In I^3R (Croft and Thompson, 1987), they used only two levels: important or default. Harper (1980) used 5 levels of importance, and Croft and Das (1989) used 4 levels. We calculate the scores of terms using the level (depth) of a node in the UIH instead of explicitly asking the user.

The significance of a term match can be measured by estimating the probability, $P(D_{t_i})$, of matching term t_i at depth (level) D_{t_i} in the UIH. $P(D_{t_i})$ is the probability of a level in a UIH. A term that matches more specific interests (deeper in the UIH) has a lower $P(D_{t_i})$ of occurring. Lower probability indicates the matching term, t_i, is more significant. The probability is estimated by:

$$P(D_{t_i}) = \frac{\text{number of distinct terms at depth } D_{t_i} \text{ in the UIH}}{n_u}$$

5.2.1.2. Length of a Term

Longer terms (phrases) are more specific than shorter ones. If a web page contains a long search term typed in by a user, the web page is more likely what the user was looking for.

In general, there are fewer long terms than short terms. To measure the significance of a term match, the probability, $P(L_i)$, of matching term t_i of length L_i in the UIH is calculated. L_i is defined as MIN(10, the length of a term). We group the longer (greater than 10) phrases into one bin because they are rare. Longer terms has a smaller probability, $P(L_i)$, of occurring, which indicates a more significant match. The probability is estimated by:

$$P(L_{t_i}) = \frac{\text{number of distinct terms of length } L_{t_i} \text{ in the UIH}}{n_u}$$

5.2.1.3. Frequency of a Term

More frequent terms are more significant/important than less frequent terms. Frequent terms are often used for document clustering or information retrieval (van Rijsbergen, 1979). A document that contains a search term many times will be more related to a user's interest than a document that has the term only once.

We estimate the probability, $P(F_i)$, of a matching term t_i at frequency F_i in a web page to measure the significance of the term. However, in general, frequent terms have a lower probability of occurring. For example, in a web page most of the terms (without the terms in a stop list (Rasmussen, 1992)) will occur once, some terms happen twice, and fewer terms repeat three times or more. Lower probabilities, $P(F_i)$, of a term t_i indicates the

significance of a term. The probability is estimated by:

$$P(F_{t_i}) = \frac{\text{number of distinct terms with frequency } F_{t_i} \text{ in a web page}}{n_t}$$

5.2.1.4. Emphasis of a Term

Some terms have different formatting (HTML tags) such as title, bold, or italic. These specially-formatted terms have more emphasis in the page than those that are formatted normally. A document that emphasize a search term as a bold format will be more related to the search term than a document that has the term in a normal format without emphasis. If a term is emphasized by the use of two or more types of special formatting we assign a priority in the order of title, bold, and italic.

The significance for each type of format is estimated based on the probability, $P(E_{t_i})$, of matching term t_i with the format type E_{t_i} in a web page. Those format types are more significant/important if the format type has lower probability of occurring in a web page. Lower probability $P(E_{t_i})$ of a matching term, t_i, indicates the term is more significant. The probability is estimated by:

$$P(E_{t_i}) = \frac{\text{number of distinct terms with emphasis } E_{t_i} \text{ in a web page}}{n_t}$$

5.2.2. Scoring a Term

5.2.2.1. Uniform Scoring

$P(D_{t_i}, L_{t_i}, F_{t_i}, E_{t_i})$ is the joint probability of all four characteristics occurring in term t_i -- D_{t_i} is the depth of a node where a term belongs to, L_{t_i} is the length of a term, F_{t_i} is the

frequency of a term, and E_{t_i} is the emphasis of a term. Assuming independence among the four characteristics, we estimate:

$$P(D_{t_i}, L_{t_i}, F_{t_i}, E_{t_i}) = P(D_{t_i}) \times P(L_{t_i}) \times P(F_{t_i}) \times P(E_{t_i})$$

The corresponding log likelihood is:

$$\log P(D_{t_i}, L_{t_i}, F_{t_i}, E_{t_i}) = \log P(D_{t_i}) + \log P(L_{t_i}) + \log P(F_{t_i}) + \log P(E_{t_i}) \quad \text{Eq. 1}$$

Smaller log likelihood means the term match is more significant. In information theory (Mitchell et al., 1997), $-\log_2 P(e)$ is the number of bits needed to encode event e, hence using $-\log_2$, instead of log, in Eq. 1 yields the total number of bits needed to encode the four characteristics. The uniform term scoring (US) function for a personalized term score is formulated as:

$$S_{t_i} = -\log_2 P(D_{t_i}) - \log_2 P(L_{t_i}) - \log_2 P(F_{t_i}) - \log_2 P(E_{t_i}) \quad \text{Eq. 2}$$

which we use as a score for term t_i. Larger S_{t_i} means the term match is more significant.

5.2.2.2. Weighted Scoring

The uniform term scoring function uses uniform weights for each characteristic. It is possible that some characteristics are more important than the others. For instance, the depth of a node (D) may be more significant than frequency (F). Therefore, we attempted to differentiate the weights for each characteristic. F and E characteristics represent the relevance of a web page. Longer terms (greater L) *r*epresent a user's interest more specifically; however, longer terms do not mean that a user is more interested in that term.

Therefore, those *L*, *F*, and *E* characteristics do not fully reflect a user's interests. It is more reasonable to emphasize *D* characteristic more than other characteristics, because *D* (depth) represents the strength of a user's interests.

A simple heuristic is used in this paper that assumes the depth of a node is at least two times more important than other characteristics. Based on this heuristic, the weights w_1=0.4, w_2=0.2, w_3=0.2, and w_4=0.2 are assigned. The weighted term scoring (WS) function for a personalized term score is formulated as:

$$S_{t_i} = -w_1 \log_2 P(D_{t_i}) - w_2 \log_2 P(L_{t_i}) - w_3 \log_2 P(F_{t_i}) - w_4 \log_2 P(E_{t_i}) \qquad \text{Eq. 3}$$

5.2.3. Scoring a Page

The *personal* page score is based on the number of interesting terms and how interesting the terms are in a web page. If there are many terms in a web page that are interesting to a user, it will be more interesting to the user than a web page that has fewer interesting terms. If there are terms in a pages that are more interesting to a user, the web page will be more interesting to the user than a web page that has less interesting terms.

The personalized page scoring function for a web page S_{p_j} adds all the scores of the terms in the web page and can be formulated as:

$$S_{p_j} = \sum_{i=1}^{m} S_{t_i} \qquad \text{Eq. 4}$$

where *m* is the total number of matching terms in a web page and S_{t_i} is the score for each distinct term. The time complexity of scoring a page is $O(n_t)$, where n_t is the number of "distinct" terms in a web page. *D* and *L* characteristics can be calculated during the

preprocessing stage of building a UIH. *F* and *L* characteristics can be calculated while extracting distinct terms from a web page.

5.2.4. Incorporating Public Page Score

Personal page scoring is not sufficient for some search engines. The success of using *public* scoring in popular search engines, such as Google's PageRank, indicates the importance of using a public page-popularity measure to determine what page a user is interested in. Many existing methods determine the public popularity of a page by determining the number pages that link to it (Haveliwala, 2002; Jeh and Widom, 2003). Many collaborative filtering approaches also use the popularity of a web page for recommendation (Eirinaki et al., 2004; Kim et al., 2004). Section 6.2.3 described our personal web page scoring function. We wish to incorporate the *public* scoring into our page scoring function so both the popularity of a page and individual interests are taken into account. We use the rank order returned by Google as our *public* score. $GOOGLE_{p_j}$ is the score of a web page p_j based on the page rank returned by Google for a search term. Google's *public* page scoring function has been found in a recent study (Delaney, 2004) to be very effective at returning pages that users find interesting. The use of Google's page rank as a *public* page score makes our experimental comparison with Google clearer, because any improvement in the ordering is due to the contribution of our *personal* page score. For a given web page, p_j, the *personal* and *public* page score (*PPS*) equation can be written as:

$$PPSp_j = c \times R(Sp_j) + (1-c) \times R(GOOGLEp_j) \qquad \text{Eq. 5}$$

where function $R(GOOGLE_{p_j})$ return the rank of a web page, p_j, with the *public* page score of $GOOGLE_{p_j}$, and $R(S_{p_j})$ is the rank of a web page, p_j, with the *personal* page score, S_{p_j}. If the function R returns the rank in an ascending order, more interesting web pages will have lower *PPS* values. Therefore, the function R reverses the rank. The *personal* page score and the *public* page score are weighted by the value of the constant c. In this paper, both functions are weighed equally: $c = 0.5$.

5.3. Experiments

In our experiments data were collected from 11 different users. Of the 11 human subjects, 4 were undergraduate students and 7 were graduate students. In terms of major, 7 were Computer Sciences, 2 were Aeronautical Sciences, 1 was Chemical Engineering, and 1 was Marine Biology. We asked each volunteer to submit 2 search terms that can contain any Boolean operators. Some examples of the search terms used are

{review forum +"scratch remover", cpu benchmark, aeronautical, Free cross-stitch scenic patterns, neural networks tutorial, DMC(digital media center), artificial intelligence, etc.}

Then, we used Google to retrieve 100 related web pages for each search term. Those collected web pages were classified/labeled by user based on two categories: *interest* and *potential interest*. The data set for *interest* has more authority because it indicates direct relevance to the current search query. The data set for *potential interest* reflects the user's general personal interests, which might not be directly relevant at the time of query. The areas of a user's *potential* interests often go beyond the boundary of a search term's specific meaning. Sometimes users find interesting web pages while searching for different

subjects. These unexpected results help the user as well. Therefore, it is also a contribution if a method shows higher precision in finding *potentially* interesting web pages.

In order to build UIHs, we also requested each volunteer to submit the web pages in their bookmarks. If there were fewer than 50 web pages in their bookmark, we asked them to collect more pages up to around 50. The minimum number of web pages was 38 and the maximum number was 72. Web pages from both bookmarks and Google were parsed to retrieve only texts. The terms (words and phrases) in the web pages are stemmed and filtered through the stop list (Rasmussen, 1992). A phrase-finding algorithm (Kim and Chan, 2004) was used to collect variable-length phrases. Words in selection boxes/menus were also removed because they did not appear on the screen until a user clicks on them. Unimportant contexts such as comments and style were also removed. Web pages that contain non-text (e.g., ".pdf" files, image files, etc.) were excluded because we are handling only text. To remove any negative bias to Google, broken links that were still ranked high erroneously by Google were excluded from the test, since those web pages will be scored "Poor" by the user for sure. The data used in this study is accessible at http://cs.fit.edu/~hkim/dissertation/dissertation.htm. Microsoft .NET language was used, and the program ran on an Intel Pentium 4 CPU.

We attempted to remove any negative bias to Google. Those web pages that contain non-text (e.g., ".pdf" files, image files, etc.) were excluded because we are handling only texts. Furthermore, the broken links that were still ranked high erroneously by Google were excluded from the test, since those web pages will be scored "Poor" by user for sure.

We categorized the *interest* as "Good", "Fair", and "Poor"; the *potential interest* is categorized as "Yes" and "No". A web page was scored as "Good", "Fair", and "Poor" depending on each individual's subjective opinion based on the definition of *interest*. It was also marked as "Yes" or "No" based on the user's *potential interest*. We evaluated a ranking method based on how many interesting (categorized as "Good") or *potentially* interesting web pages (categorized as "Yes") the method collected within a certain number of top links (Bharat and Mihaila, 2001) (called "Top link analysis"). It is realistic in a sense many information retrieval systems are interested in the top 10 or 20 groups. Precision/recall graph (van Rijsbergen, 1979) is used for evaluation as well (called "precision/recall analysis"). It is one of the most common evaluation methods in information retrieval. However, traditional precision/recall graphs are very sensitive to the initial rank positions and evaluate entire rankings (Croft and Das, 1989). The formula for precision and recall were:

```
Precision = Number of "Good" or "Yes" pages retrieved in the
        set / Size of the set
Recall = Number of "Good" or "Yes" pages retrieved in the set /
        Number of "Good" or "Yes" pages in the set
```

where the "set" is the group of top ranked web pages. In this paper we study five groups: Top 1, 5, 10, 15, and 20.

5.4. Results and Analysis

We compare four ranking methods: Google, Random, US, and WS. Google is the ranking provided by Google. Random arbitrarily ranks the web pages. US and WS are the two proposed methods based on a personal UIH learned from a user's bookmarks. For

Random, US, and WS, the top 100 pages retrieved by Google are re-ranked based on the method. Each method is analyzed with two data sets: a set of web pages chosen as interesting and another chosen as potentially interesting by the users. Top link analysis, precision/recall analysis, the sensitivity of personal score weight *c* (Section 6.2.4) are discussed.

5.4.1. Interesting Web Page

5.4.1.1. Top Link Analysis

Web search engine users are usually interested in the links ranked within top 20 (Chen and Sycara, 1998). We compare each method only with Top 1, 5, 10, 15, and 20 links on the *interesting* web page data set and present the results in Table 20. The first column is the methods; the next five columns present the precision values of each method with respect to the five Top links. The values in each cell are the average of 22 search terms' precision values. High precision value indicates high accuracy/performance. Precision values higher than Google's are formatted as bold and the percentage of improvement is within parentheses. The highest precision value in each column is underscored.

The results show that our WS method was more accurate than Google in three Top links (Top 10, 15, and 20) and the percentages of improvements are at least 13%, while WS ties with Google for Top 1 and Top 5. In terms of the highest precision, WS showed highest performance in four columns; Google showed in only two columns and the values are equal to WS. Compared to US, WS showed higher precision in four (Top 1, 5, 15 and 20) of the five columns. Random was the lowest as we expected, showing the lowest

precisions in all five columns. These results indicate that WS achieves the highest overall precision.

We also wanted to know which search terms yielded higher precision with WS than with Google and analyzed the precision with respect to each individual search terms. Out of 22 search terms (11 users × 2 search terms), WS achieved higher precision for 12 search terms (55%), Google did for 8 search terms (36%), and they were even for 2 search terms (9%). Since the UIH is built from a user's bookmarks, we analyse the bookmarks to understand the search terms that did not perform well using WS. When we compare the bookmarks with the "good" retrieved web pages, we found that they are unrelated. For example, a volunteer used "woodworking tutorial" as a search term, but he never bookmarked web pages related to that term. This implies bookmarks are useful for building user profiles, but they are not sufficient. We will discuss enhancements in the conclusion.

5.4.1.2. Statistical Significance

In order to see if this improvement is statistically significant we conducted t-Test (paired two samples for means) between two groups of individual search terms with Google and WS for each Top link. There was no statistically significant difference between WS and Google for any Top link with 95% confidence (P=1and t=2.079 for Top 1; P=1and t=2.079 for Top 5; P=0.328 and t=2.079 for Top 10; P=0.204 and t=2.079 for Top 15; P=0.147 and t=2.079 for Top 20).

To understand why our improvements are not statistically significant, we analyze the variance in the precision values. In Figure 22 we plot the average and the standard

deviation (SDs) of 22 search terms' precisions from Google with respect to the five Top links. The *x*-axis shows the Top links and *y*-axis represents the average and the SD of precision values. The dots in the middle of vertical bars are the averages and the bars themselves represent the SD values. Variance was large for Top 1 and decreases when more links were considered.

To understand the difficulty of improving Google's ranking, we calculate the number of multiples needed to achieve one SD from the average. Formally, the number of multiples is defined as (Avg.+SD)/Avg. The larger the number of multiples indicates more difficulty in beating Google with statistically significance. The number of multiples for Top 1 is 3.35, Top 5 is 2.79, Top 10 is 2.72, Top 15 is 2.71, and Top 20 is 2.72. In information retrieval doubling or tripling the precision for a large variance like Google's is rare. From our calculated number of multiples, we need to at least double or triple the precision to achieve statistically significant improvement over Google's ranking.

To further demonstrate the difficulty, we applied the same t-Test to precision values from Google and Random (P=0.134, t=2.079 for Top 1; P=0.179, t=2.079 for Top 5; P=0.062, t=2.079 for Top 10; P=0.035, t=2.079 for Top 15; P=0.024, t=2.079 for Top 20). We found that, though Google's improvement over random is statistically significant for Top 15 and 20, it is *not* statistically significant for Top 1, 5, and 10.

5.4.1.3. Precision/Recall Analysis

Precision/recall analysis visualizes the performance of each method in graphs as shown in Figure 23. The *x*-axis is recall and *y*-axis is precision. The line closer to the upper-right corner has higher performance. WS and US are closer to the upper-right corner

than Google except with recall values lower than .15 (after Top 5). In general, WS outperforms US and Random.

5.4.1.4. Varying Personal Weight

The performance of WS may depend on how much we weigh the *personal* page score over the *public* page score. The parameter c in Section 6.2.4 represents the weight for the *personal* page score. For example, c=0.9 means the page is scored by 90% of *personal* page score and 10% of *public* page score. We experimented with c={0.9, 0.7, 0.5, 0.3, and 0.1} and measured the corresponding precision and recall. The results are plotted in Figure 24 and each line represents a different c value. The line closer to the upper right corner indicates higher performance. c=0.9 has lowest precision. c=0.1 achieved the second lowest precision except for recall values lower than 0.2. Figure 25 enlarges the scale of recall between 0 through 0.2 in Figure 24. It is still not clear which one is higher than the others except the line with c=0.9; however, c=0.5 in general seems to show the highest performance. Therefore, we chose c=0.5 as the weight of *personal* page score.

Table 20. Precision in Top 1, 5, 10, 15 and 20 for *interesting* web pages

	Top 1	Top 5	Top 10	Top 15	Top 20
Google	.36	.34	.277	.285	.270
Random	.14	.25	.205	.206	.209
US	.32	.31	**.323 (17%)**	**.315 (11%)**	**.305 (13%)**
WS	.36	.34	**.314 (13%)**	**.327 (15%)**	**.309 (14%)**

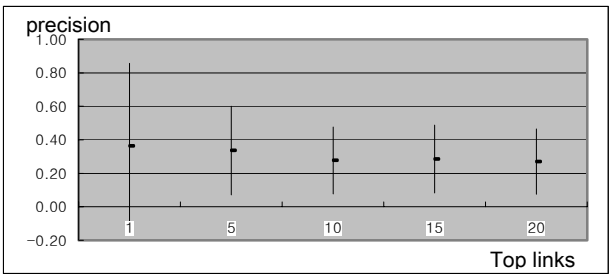


Figure 22. Average and SD of precision with Google

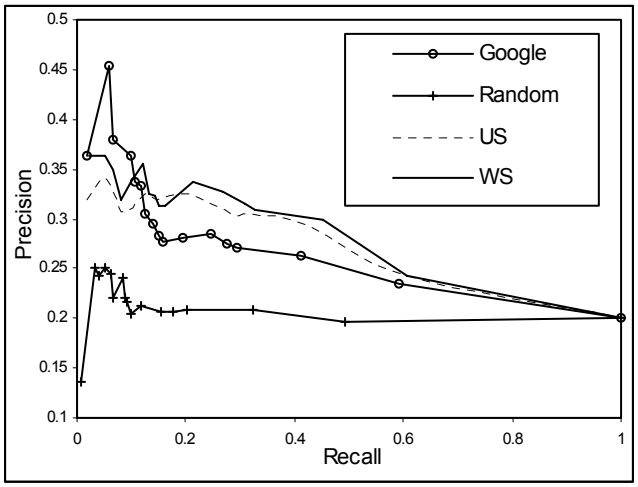


Figure 23. Precision/recall graph for *interesting* web pages

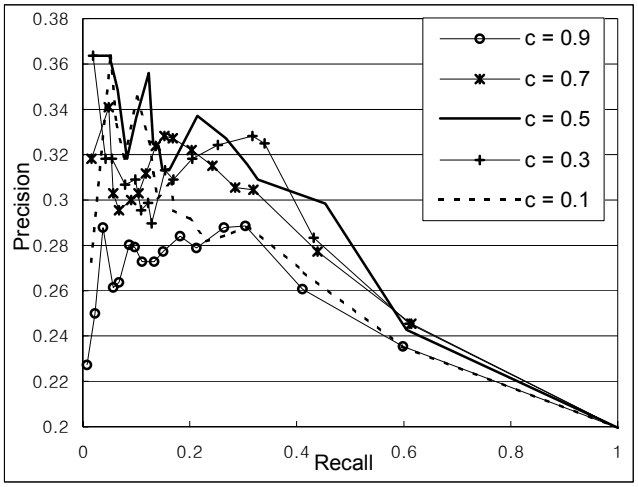


Figure 24. Precision/recall with personal score weight *c*

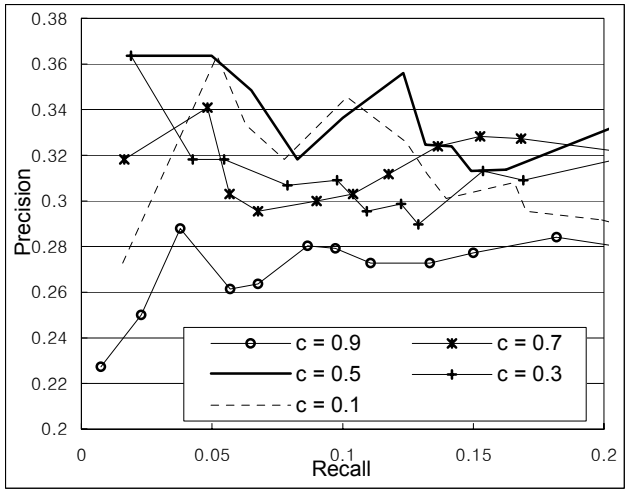


Figure 25. Up to 20% recall of Figure 24

5.4.2. Potentially Interesting Web Page

5.4.2.1. Top Link Analysis

We compare our four methods with Top 1, 5, 10, 15, and 20 links on the *potentially* interesting web page data set and present the results in Table 21. The values in each cell are the average of 22 search terms' precision values. The ways of reading this table and the table for *interesting* web pages are similar.

WS showed higher performances than Google in all five Top links. All five precisions achieved by WS are the highest values as well. The percentages of improvements are between 3% and 17%. Random showed the lowest in all five Top links. The reason for the improvement of WS is, we predict, because the UIH that was derived from a user's bookmarks supported the user's *potential* interest. It might be difficult for Google that used the *global*/*public* interest to predict individual user's broad *potential* interests.

We also counted what search terms yielded higher precision with WS than with Google. WS achieved higher performance for 12 search terms (55%), Google made for 8 search terms (36%), and they were even for 2 search terms (9%) out of 22 search terms. The reason for the low performance of some search terms might be because there is no relation between his/her bookmarks and the search terms.

5.4.2.2. Statistical Significance

The t-Test between the two groups of 22 individual search terms with WS and Google showed no statistically significant difference with 95% confidence for any Top link

(*P*=0.665 and *t*=2.079 for Top 1; *P*=0.115 and *t*=2079 for Top 5; *P*=0.466 and *t*=2.079 for Top 10; *P*=0.580 and *t*=2.079 for Top 15; *P*=0.347 and *t*=2.079 for Top 20).

We analyze the variance in the precision values to understand why the improvements are not statistically significant. The graph in Figure 26 illustrates the average and the standard deviation (SD) of 22 search terms' precisions with Google to the five Top links. Variance was large for Top 1 and decreased as more links were considered.

In order to estimate how difficult it is to achieve statistically significant improvements over Google, we calculate the number of multiples, which is (Avg.+SD)/Avg. The number of multiples for Top 1 is 2.77, for Top 5 is 2.53, for Top 10 is 2.59, for Top 15 is 2.58, for Top 20 is 2.60. From our calculated number of multiples, we need to at least double the precision for achieving statistically significant improvement on *potentially* interesting web pages.

5.4.2.3. Precision/Recall Analysis

The results from precision/recall graph for *potentially* interesting web pages in Figure 27 and the Top link analysis in Table 21 are similar. WS was closer to the upper-right corner than Google, US, and Random over all. WS outperformed other methods on *potentially* interesting web pages data set.

Table 21. Precision in Top 1, 5, 10, 15 and 20 for *potentially interesting* web pages

	Top 1	Top 5	Top 10	Top 15	Top 20
Google	.59	.53	.514	.509	.475
Random	.36	.39	.350	.358	.364
US	.59	**.58 (9%)**	**.536 (4%)**	**.521 (2%)**	**.493 (4%)**
WS	**.64 (8%)**	**.62 (17%)**	**.541 (5%)**	**.524 (3%)**	**.498 (5%)**

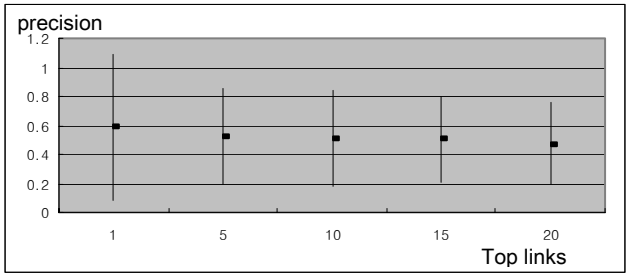


Figure 26. Average and SD of precision with Google

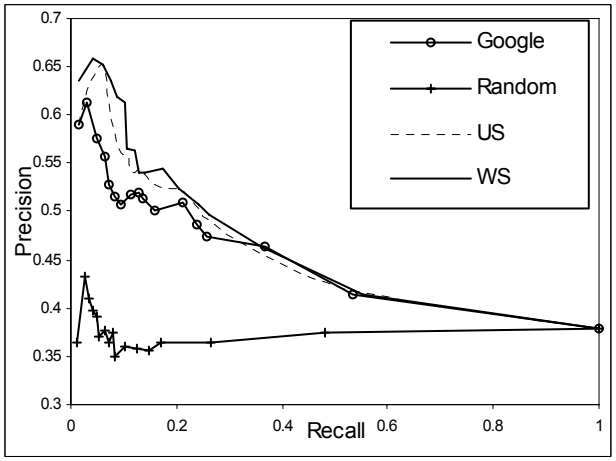


Figure 27. Precision/recall graph for *potentially interesting* web pages

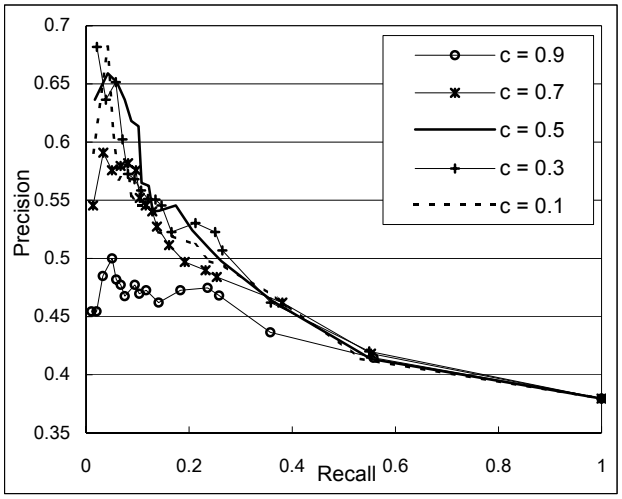


Figure 28. Precision/recall with personal score weight *c*

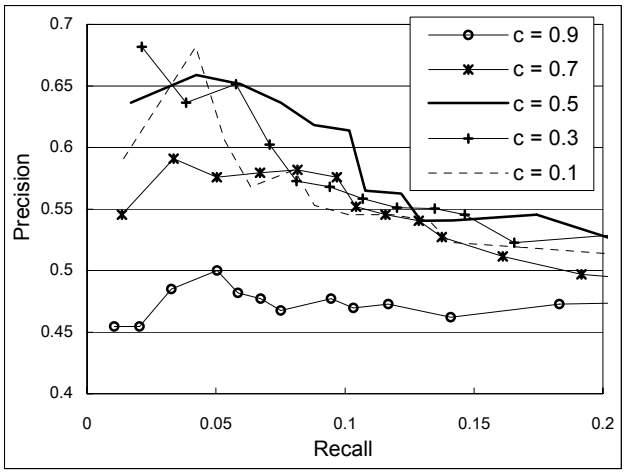


Figure 29. Up to 20% recall of Figure 28

5.4.2.4. Varying Personal Weight

In order to see the improvement of WC's performance with different *personal* weights, we varied the parameter for the weight: c = {0.9, 0.7, 0.5, 0.3, and 0.1}. The results are plotted in Figure 28. c=0.9 draw the lowest line; c=0.7 looks the second lowest line. Figure 29 enlarges the scale of recall in Figure 28. It looks clear that c=0.5 in general seems to show the highest performance. Therefore, we chose c=0.5 for the weight of *personal* page score. The weights for personal page score with both data sets are the same.

5.5. Summary

The purpose of this research is to devise a new method of ranking web search results to serve each individual user's interests. A user profile called UIH is learned from his/her bookmarks. For scoring a term in a web page that matches a term in the UIH, we identified four characteristics: the depth of tree node in the UIH that contains the term, the length of the term, the frequency of the term in the web page, and the html formatting used for emphasis. Our approach uses the terms filtered though stop list in web pages (Rasmussen, 1992). This approach removes the process of selecting important/significant terms unlike other information retrieval techniques (Pazzani and Billsus, 1997). Therefore, we can handle smaller data set and reduce the danger of eliminating new important terms. We evaluated methods based on how many interesting web pages or potentially interesting web pages each algorithm found within certain number of top links (Bharat and Mihaila, 2001). Traditional precision/recall graphs (van Rijsbergen, 1979) were also used for evaluation. We counted which search term showed higher performances with WS than with Google as well.

We compared four ranking methods: Google, Random, US, and WS. Google is the most popular search engine and posts the best ordering results currently. Random method was chosen to see the improved performance of Google and our new methods. We used two data sets: interesting web pages that are relevant to the user search term and potentially interesting web pages that could be relevant in the future. On interesting web pages, the Top link analysis indicated WS achieved at least 13% higher precision than Google for Top 10, 15 and 20 links on average. WS outperformed US and Random in general also. The precision/recall analysis showed that WS outperformed Google except with recall values lower than .15. Out of 22 search terms, WS achieved higher precision than Google for 12 search terms (55%). On potentially interesting web pages, WS achieved the highest performance in all five Top links with improvement over Google between 3% and 17%. It also outperformed the other methods in the precision/recall graph. The analysis of individual search terms yielded the same results as on interesting web pages. A weight of 0.5 for the personal ranking seemed to show the highest performance on both data sets. Therefore, these results conclude that WS can provide more accurate ranking than Google on average. The improvement of WS was not statistically significant because the precision values of Google had large variance. The reason for the low performance of some search terms might be because there is no relation between his/her bookmarks and the search terms. We may be able to relieve this problem by incorporating interesting web pages based on implicit interest indicators such as mouse movements (Kim and Chan, 2005) in addition to bookmarking.

During the experiment, we observed that users do not tend to measure index pages as "Good". It is because index pages usually contain long lists of hyperlinks with little

description for a user to find interesting. To identify index pages automatically, we count the number of "outside words" (the text outside anchor tags), which usually provide the subject content. However, our approach of penalizing the index pages did not make much improvement in our initial experiments. We will examine this approach further in the future.

Measuring the precision with clustered search results like the results from Vivisimo (2005) may show different performance from Google's. In a clustered search engine, a link that does not belong to the top 10 in whole can belong to the top 10 in some sub clusters. The clustered search results provide users easier access to the *interesting* links after Top 10 or 20. Since WS showed higher performance for those links than Google as shown in Section 6.4.1.3, we assume that our method may get higher performance with clustered search engines. We may be able to make a search engine more interactive using a UIH. For example when a user's query resides in an intermediate node in his/her UIH, we can ask a user to choose more specific interests providing the terms in the child nodes, or in another sub-trees in the UIH.

Chapter 6

Implicit Indicators for Interesting Web Pages

A user's interest in a web page can be estimated by unobtrusively (implicitly) observing his or her behaviour rather than asking for feedback directly (explicitly). Implicit methods are naturally less accurate than explicit methods, but they do not waste a user's time or effort. Implicit indicators of a user's interests can also be used to create models that change with a user's interests over time. Research has shown that a user's behaviour is related to his/her interest in a web page. We evaluate previously studied implicit indicators and examine the time spent on a page in more detail. For example, we observe whether a user is really looking at the monitor when we measure the time spent on a web page. Our results indicate that the duration is related to a user's interest of a web page regardless a user's attention to the web page. The thicker features in Figure 30 describe the position of this work in our overall diagram of web personalization.

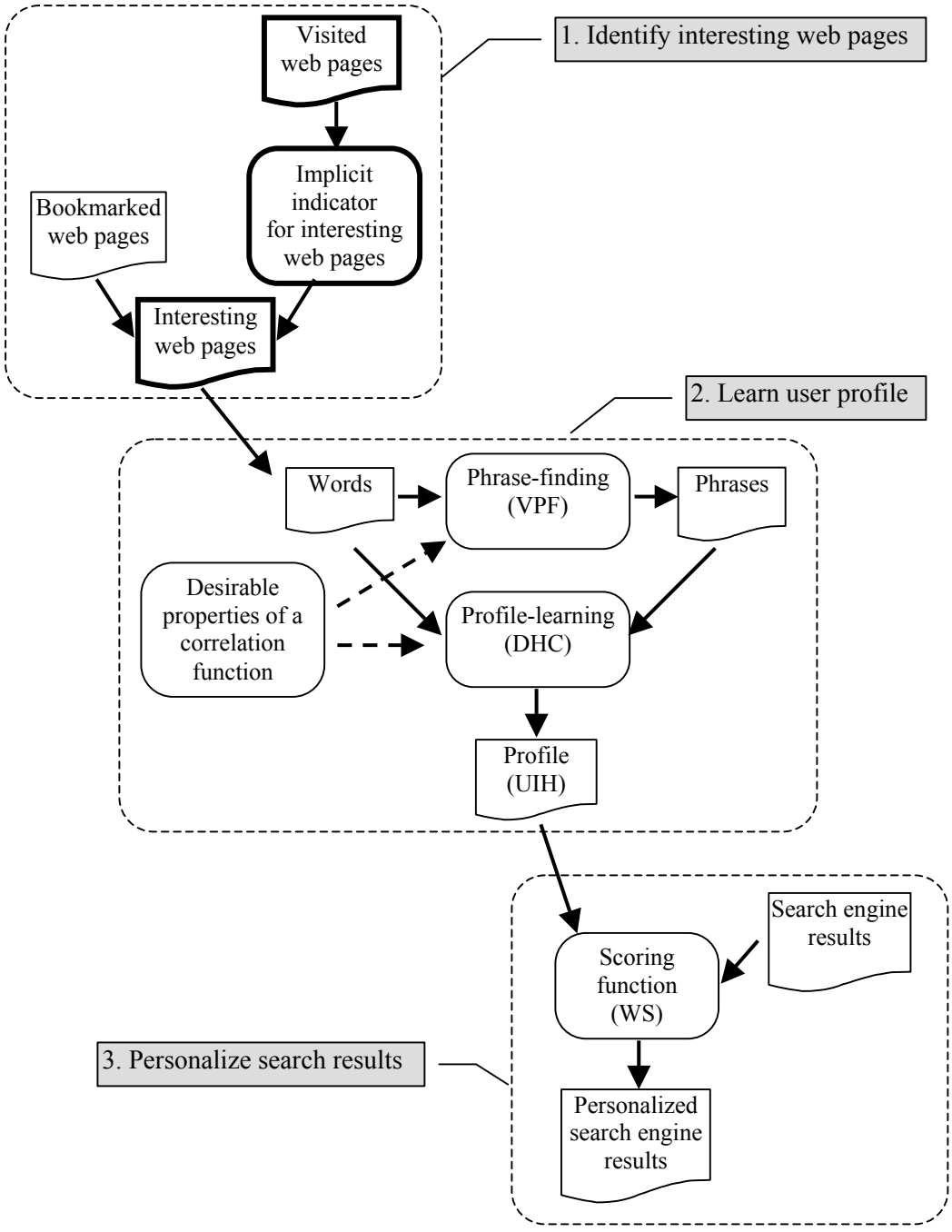


Figure 30. Implicit indicator for interesting web pages

6.1. Implicit Interest Indicators

The time spent on a web page is one of the most intuitive candidates for user interest indicators. This paper thoroughly examines whether duration is related to a user's interest. This section describes duration, as well as other user interest indicators that will be examined. The reason why each indicator is chosen is explained and how each indicator is measured is described.

6.1.1. Complete Duration

A user may tend to spend more time on pages that he or she finds interesting, so we record the duration spent on a web page. The *complete duration* is defined as the time interval between the time a user opens and leaves a web page. Some web pages contain many images that delay the downloading time, so we start measuring the duration after the entire page is loaded. Thus, the *complete duration* won't be affected by the connection speed, the amount of Internet traffic, or the CPU speed. The *complete duration* for a web page can be calculated by subtracting the time of finishing downloading the current web page from the time of leaving the web page. The *complete duration* is different from the duration used by Jung (2001). His duration includes the downloading time of a web page.

6.1.2. Active Window Duration

Most modern operating systems allow a user to multitask, or run several applications at the same time. A user may write a report or chat while browsing a web page. Those other applications can be unrelated to the contents of a web page. If a user

spent one hour writing a homework paper with a web browser minimized, the *complete duration* of the web page could be one hour. This is very likely to provide erroneous indications of user interest. In order to avoid being affected by this problem, we determine whether a web browser is active or not. The time that a web browser is inactive is subtracted from the *complete duration*. We call this duration *active window duration* since we count the time only when a web browser is active.

6.1.3. Look At It Duration

Users are not always reading a web page when the web browser is active. They can easily be talking to friends or having a coffee break, while the web browser is active. The *active window duration* can easily be more than 30 minutes if a user leaves the browser active and goes for a coffee break. We may be able to detect the user's absence by detecting the action of mouse movement. However, a better solution is to use a camera that detects a user's face orientation. A camera can even check if a user is looking at the web browser or if his attention is diverted. This duration will be more accurate than the *active window duration* in terms of checking user's attention to a web page. Since this duration counts the time that a user is looking at the web browser, we call it *look at it duration*. The *look at it duration* can be calculated by subtracting the time when a user does not look at the browser from *active window duration*.

6.1.4. Distance of Mouse Movement

Many people move their mouse while reading the contents of a web page. Mouse movement can occur while looking at an interesting image, or when pointing at interesting

objects. We hypothesize that the more distance a mouse moves, the more a user be interested in the web page. This indicator was also examined by Jung (2001). Our distance is a little bit different from his in a sense of detecting overall mouse movement. He counted on the mouse movement only when the mouse point is inside the active browser. The *distance of mouse movement* is detected by its *x* and *y* coordinates on a monitor every 100 milliseconds. The formula is

$$mouse\_movement(pixels) = \sum_{i=1}^{t-1} Dist(P(t_i) - P(t_{i-1}))$$

where time *t* is the *active window duration*, the time interval, t_i-t_{i-1}, is 100 milliseconds, $P(t_i)$ is a mouse location with *x* and *y* coordinates at time t_i, and the *Dist* function is a Euclidean distance.

6.1.5. Number of Mouse Clicks

People use "click" to hyperlink to another web page. In addition, clicking can be considered as a habitual behaviour (Jung, 2001). Clicking can be a way of expressing our emotions such as if some people are happy to find a product that they were looking for (e.g., book), then they can click the object several times repeatedly. This indicator was examined in Kixbrowser (Jung, 2001), Curious browser (Claypool et al., 2001), Goeck's browser (Goecks et al., 2000), and Letizia (Liberman, 1995). We use the hypothesis that the greater the *number of mouse clicks* on a web page is, the more a user is interested in it (Jung, 2001). The *number of mouse clicks* is counted every time a mouse button is clicked.

6.1.6. Distance of Scrollbar Movement

A user can also scroll a web page up and down by dragging a scrollbar. Those dragging events can occur several times while a user is reading a web page. The *distance of scrollbar movement* for an occasion, E, can be calculated by measuring the mouse movement every 100 milliseconds. By summing all distances of scrollbar movement for all occasions, the *distance of a scrollbar movement* for a web page can be calculated. The formula is

$$scrollbar\_movement(pixels) = \sum_{j}^{E} \sum_{i=1}^{E(j)-1} \left| P(t_i) - P(t_{i-1}) \right|$$

where E is the number of times the scrollbar is pressed, time $E(j)$ is the duration that the scrollbar is dragged in a single dragging event, and t_i-t_{i-1}, is 100 milliseconds. We hypothesize that greater scrollbar movement is correlated with more user interest in a web page.

6.1.7. Number of Scrollbar Clicks

The length of many web pages is longer than the height of a monitor. If a user finds a web page interesting, he or she may read further down the web page. A user can scroll down a web page either by clicking or by dragging the scrollbar. Those events are counted separately. The number of scrollbar clicks is counted every time a user clicks scrollbar. As a user scrolls a web page up and down by clicking, the number of scrollbar clicks increases. Jung (2001), Goecks et al. (2000), and Claypool et al. (2001) measured this event and reported that it is a good indicator. We hypothesize that we will also find that the number of scrollbar clicks is correlated with a user's interest in the web page.

6.1.8. Number of Key UP and Down

When scrolling a web page, some people use the “up” and “down” keys instead of the scrollbar. This indicator is similar to the number of scrollbar clicks and the distance of scrollbar movement. The hypothesis is that the greater the *number of key up and down* presses, the more a user is interested in the web page. This event is measured by increasing the count every time a user strikes up or down keys. Curious browser (Claypool et al., 2001) and Jung (2001) measured keyboard activities. But they did not measure the key up and down for measuring scrollbar movement.

6.1.9. Size of Highlighting Text

While reading a web page, if a user copies some contents of the web page it probably means that the user is interested in the web page. Furthermore, a user can also habitually highlight portions of the page that they are interested in, which is a sign that the user is interested in the page. We assume that the more a user highlights in a web page, the more a user is interested in that web page. A user can highlight several different sentences in a web page for several different occasions. We sum all highlighted contents at the end. Jung (2001) examined this indicator. He used the Euclidean distance between two points of pressing and releasing. The weakness of his measure resides in neglecting the texts highlighted horizontally when the mouse moves vertically. In order to solve this problem, we assumed a character is 5 pixels, each line has 80 characters, and distance between two lines is 20 pixels on average. The formula is

$$highlighting\_text = \sum_{j}^{E} DistY_j / 20 \times 80 + DistX_j / 5$$

where *E* is the number of occasions when highlighting occurs, *DistY* is the vertical distance between two points, and *DistX* is the horizontal distance between two points.

6.1.10. Other Indicators

We also measure other less-frequently-used events such as *bookmark*, *save*, *print*, and *memo*. A user usually bookmarks web pages in order to visit them later again. We assume those bookmarked web pages are interesting to a user (Li et al., 1999; Maarek and Ben-Shaul, 1996). This can be measured by detecting bookmarking activities during the experiment. Users save important/interesting web pages in their hard drive by using the "Save As" command. This also implies that those saved web pages are interesting to users (Liberman, 1995). This indicator is also counted by detecting saving activities during the users' browsing. Most web browsers allow users to print web pages. These printed web pages are likely to be interesting to users (Kim et al., 2001). The Memo box is a new feature added in our system. It allows a user to write down a short description on a web page. When the user visits the web page again, the message shows up on the Memo box automatically. We assume that if a user is interested in a web page, then s/he will write a note about the web page.

6.2. Detecting Face Orientation

The *look at it duration* is the time when a user looks at a computer screen. In order to count the time we monitor a user's head orientation. In this chapter, we detail how we detect the head orientation using a webcam. For detecting head orientation, we use three

dots on the hat that a user wears during our experiment. Discussed are how to detect the three dots and how to learn head orientation with the three dots.

6.2.1. Detecting Three Dots

To recognize head orientation in an image, the background is usually removed (Intel Inc., 2001). The term "background" stands for a set of motionless image pixels, that is, pixels that do not belong to a human object in front of the camera. The background image can reduce the performance or increase the time complexity.

A simple model of removing background may assume that every background pixel brightness varies independently, according to normal distribution. The background characteristics can be calculated by the mean and standard deviation for every pixel of several dozens of frames collected. After that the pixel in a certain pixel location in certain frame is regarded as belonging to a moving object if condition Abs(mean(x,y)–p(x,y))>3×StandardDeviation(x,y) is met, where p(x,y) is a pixel in a new frame (Intel Inc., 2001). However, as the object moves closer to the camera (from "Far" to "Close" in Figure 31), the background color changes from dark color to almost white color -- the assumption is not right (do not follow normal distribution).

Since a user is wearing a black hat, the derivation between two adjacent points helps detect the boundary of the hat. If we continue to remove the pixels that have lower derivation than a threshold starting from edges (left, right, and top) and stop when the

derivation is higher than the threshold, at the end the object alone remains without background. The formula for the derivation is

$$p(x,y)=\begin{cases}1 \; if \; \max(Dist(p(x,y),p(x-1,y)),Dist(p(x,y),p(x,y-1))) < 200 \\ 0 \qquad\qquad\qquad\qquad otherwise\end{cases}$$

where $p(x,y)$ is the pixel at the coordination of x and y and 200 of the constant is chosen as the threshold by observation. The distance between two pixels is the difference of green color values of two pixels.

Once the background is removed, the image is converted into a black and white image – entire image becomes black except three dots. Each dot/circle is detected by connecting adjacent white pixels as shown in Figure 32. Some erroneous white pixels are cleaned by applying some rules such as removing pixels which size is smaller than 5 and choosing top three circles. The 3 dots are depicted in Figure 33: left, center, and right dots.

a) Far

b) Medium

c) Close

Figure 31. A retrieved image

Figure 32. Detected three dots

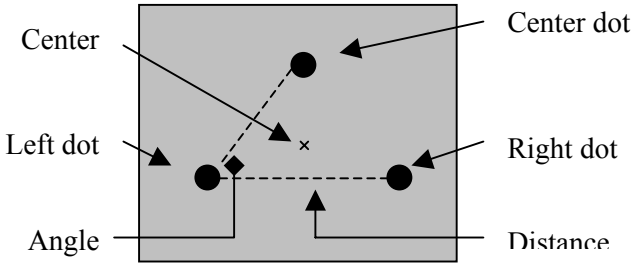


Figure 33. Three dots in an image

6.2.2. Learning Face Orientation

6.2.2.1. Input/Output Parameters

The information about the 3 dots becomes the input parameter of a learning algorithm (this will be explained later). All examined information about the three dots is

> {the coordination of left, center, right dots; the sizes of left, center, right dots; distances between dots; the coordination of center dot; the ratios between the sizes of dots; the ratios between the distances of dots; the angles of left, center, right dots; the levelness of the line between left and right dots}.

The size of a dot is the number of pixels in a dot. If the size of a left dot is smaller and the size of the right dot, then it indicates that the face is not oriented to the web browser. The ratio between the sizes of the left and the right dots can be a more accurate indicator because the ratio is independent from the distance between a face and the monitor. The distances between dots can be used to measure the face orientation. Short distance between the left and the right dots shows that the face does not direct to the monitor. The relative ratios among the three distances are independent from the distance between a face and the monitor. We also use the three angles of three dots and the levelness between the left and the right dots. The levelness will tell us whether the head is up strait or not. All input values are normalized (between 0.1 – 0.9). The output parameter is the head orientation: looking at the monitor (0.1) and not looking at the monitor (0.9).

6.2.2.2. Learning Algorithm

Artificial Neural Network (ANN) algorithm provides a robust approach to approximating various types of examples. The BACKPROPAGATION algorithm is the most commonly used ANN learning technique. The BACKPROPAGATION algorithm has proven surprisingly successful in many practical problems such as learning to recognize handwritten characters (LeCun et al., 1989), learning to recognize spoken words (Lang et al., 1990), and learning to recognize faces (Cottrell, 1990). This algorithm is appropriate for problems with the following characteristics (Mitchell, 1997): the input and the output should be able to be represented as vectors; the training examples may contain errors; long training times are acceptable; fast evaluation of the learned target function may be required; the ability of humans to understand the learned target function is not important.

Our input and output data can be represented as vectors; error can reside in our training data; training time is not important in our work; since our web browser has to evaluate an image every other seconds, fast evaluation of the learned target function is required; the ability of humans to understand the learned target function is not important. These characteristics of our example data match BACKPROPAGATION algorithm.

We used BACKPROPAGATION to learn the face orientation with 100 hidden units and 0.03 of learning rate. The results reached more than 90% accuracy. Since the purpose of this paper is not to evaluate the performance of our face-orientation-detection method, we omit the technical details and the analysis of the results.

6.3. Experiments

6.3.1. Experimental Data and Procedures

For our experiments, we built a web browser that can record the indicators described above from user's behaviour and used a camera to record images for identifying face orientation. 11 data sets were collected from 11 different users. Of the 11 human subjects, 4 were undergraduate students, 6 were graduate students, and 1 was a Ph.D. student. In terms of major, 7 were Computer Sciences, 2 were Aeronautical Sciences, 1 was Chemical Engineering, and 1 was Marine Biology. Each subject was asked to spend a total of 2 hours at the computer. Volunteers were allowed to leave the computer and do other non-computer work. All volunteers were encouraged to behave as normal as possible. To get a variety of behaviours, we asked the volunteers to divide their activities into multiple sessions, each of which does not exceed 1 hour.

In the browser used in our experiment, most of the functions in Microsoft Explore 6.0 were implemented. The popup windows were disabled initially, but our browser allowed a user to change the option to able them. We asked users to bookmark more than 10 pages, save more than 5 pages, print more than 5 pages, use Memo on more than 5 pages. The browser had Memo box so that users can write small note on a web page. Our web browser takes a picture of a user every 2 seconds. Every time a user leaved a web page, the web browser asked the user how much they are interested in the web page – there were 5 scales between "not interested" (1) and "very interested" (5).

The interests were subjective to each user. The system had a “rescore” button to allow changing the score marked in the previous visit. The browser was written in Visual Studio .NET and ran on a Pentium 4 CPU. The Operating System was Windows XP.

6.3.2. Evaluation Criteria

Two evaluation criteria are used: how accurate an indicator could predict a user’s interest and how many users an indicator can accurately predict their interests. Instead of mixing all users’ data sets together, each individual data set was analysed separately so that we could clearly observe whether some indicator predicted certain individual’s interests more accurately than other indicators. An indicator that could predict the score with a lower variance is a more accurate indicator. In order to evaluate each indicator to see which one is more predictable, we use ANOVA (Analysis of Variance). Jung (2001) treated the scale as numeric scale and applied linear regression, multiple linear regression, etc. methods. We, however, consider the interest scores as discrete values and check if the indicator values are significantly different among the five different interest scores provided by the user. For ANOVA, we use a confidence level of 95% to indicate statistical significance. If the difference is significant, indicator values can predict interest scores. As a second criterion, we count the number of users predicted accurately by an indicator. This criterion indicates how reliable the indicator is across different users.

6.4. Results and Analysis

This section analyzes the data collected from the users who participated in our experiment. There are two data sets: "visits with maximum duration" and "all visits". For web pages that a user visited more than once, the score might be the same, but all other information (the durations or number of mouse clicks etc.) may be different. The "visits with maximum duration" data set contains only page views where the user stayed for the longest period of time. The maximum duration is determined using *complete duration,* which is described in Section 3.1. The "all visits" data set contains all page views collected in our experiment. We believe that the "visits with maximum duration" data set is more useful than "all visits", because users do not tend to read the web page again if they know about a web page before (Billsus and Pazzani, 1999). On average, users had 182 visits in the "visits with maximum duration" data set, and users had 291 visits in the data set of "all visits". Jung (2001) only used the "all visits" data set.

6.4.1. Visits with Maximum Duration

Table 22 shows the experimental results with "visits with maximum duration" data set. The table summarized which indicator is reliable for which volunteer. The first column is users, the second column is *complete duration* (*Complete*), the third column is *active window duration* (*Active*), the rest columns are for *look at it duration* (*LookAtIt*), *distance of mouse movement* (*MousMove*), *number of mouse clicks* (*MousClk#*), *distance of scrollbar movement* (*ScrolMov*), *number of scrollbar clicks* (*ScrolCk#*), *number of key up and down* (*KeyUpDn#*), and *size of highlighting text* (*Highligh*). They are implicit

indicators examined. The "√" mark means that the hypothesis for the indicator is statistically significant and "x" means that it was not. The mark "?" means it was unavailable to apply statistical methods to the data due to various reasons such as limited data. The last row indicates how many users' interests can be predicted by that indicator – the number of "√" mark for each column.

The Indicators *Complete*, *Active*, *LookAtIt*, and *MousMove* were able to classify 8 users' interests towards web pages (73%). The indicator of *MousClk#* was the next best indicator, which was recognized as the best in (Jung, 2001). Indicators of *KeyUpDn#* and *Highligh* were able to distinguish the lowest number of users' interests – *KeyUpDn#* was significant to only 1 user and *Highligh* was significant to only 3 users. No indicator could predict User 5's interest. The indicator *Highligh* could predict User 7, but no other indicators could do his interest. Indicator of *ScrolMov* was also valid only to User 4. These results indicate that there was no indicator that was valid to all of the users. Depending on users, an indicator may or may not be valid.

We expected that the *LookAtIt* would be the most accurate indicator, but the result did not turn out as we expected. We suspect that this was because they did not move around much and looked at the monitor most of the time while browsing. In practice, a user can use browser longer period.

6.4.2. All Visits

Table 23 shows the experimental results with the data set of "all visits". The table summarized which indicator is reliable for which volunteer. The implicit interest indicators *Complete*, *Active*, *LookAtIt*, and *MousMove* were able to predict the interests of 7 users

(64%) that participated in the study. This means that when we used "visits with maximum duration" we could predict more number of users – 8 users. This result notifies that the "visits with maximum duration" data set is more useful in predicting users' interests more accurately than the data set of "all visits".

The indicator of *MousClk#* was the next best indicator and was able to predict the interests of 6 users. User interest was more accurately predicted by the *MousClk#* implicit indicator in the "all visits" data set, but this was less predictable than the 4 indicators above. This result is similar to the findings of Jung (2001), who also used the "all visits" data set, and where *MouseClk#* was found to be the best indicator. No indicator could predict User 5's interest. User 4's interest could be predicted only by *ScrolCk#* and User 7's interest could be predicted only by *Highligh*. These results also indicate that different indicators can predict different people.

Table 22. ANOVA test with "visits with maximum duration" data set

Users	Complete	Active	LookAtIt	MousMove	MousClk#	ScrolMov	ScrolCk#	KeyUpDn#	Highligh
User 1	√	√	√	√	√	×	×	?	×
User 2	√	√	√	√	√	√	√	√	√
User 3	√	√	√	√	√	√	√	?	√
User 4	×	×	×	×	×	√	×	?	×
User 5	×	×	×	×	×	×	×	?	×
User 6	√	√	√	√	×	×	√	×	×
User 7	×	×	×	×	×	×	×	×	√
User 8	√	√	√	√	√	×	×	×	×
User 9	√	√	√	√	×	×	×	×	×
User 10	√	√	√	√	√	√	√	×	×
User 11	√	√	√	√	×	×	×	×	×
Sum	**8**	**8**	**8**	**8**	**5**	**4**	**4**	**1**	**3**

Table 23. ANOVA test with the data set of "all visits"

Users	Complete	Active	LookAtIt	MousMove	MousClk#	ScrolMov	ScrolCk#	KeyUpDn#	Highligh
User 1	√	√	√	√	√	√	√	?	×
User 2	√	√	√	√	√	×	√	√	×
User 3	√	√	√	√	√	√	√	?	√
User 4	×	×	×	×	×	×	√	×	×
User 5	×	×	×	×	×	×	×	×	×
User 6	√	√	√	√	√	×	√	×	×
User 7	×	×	×	×	×	×	×	×	√
User 8	√	√	√	√	√	×	×	×	√
User 9	√	√	√	√	√	×	×	×	×
User 10	×	×	×	×	×	√	×	√	×
User 11	√	√	√	√	×	√	×	√	×
Sum	**7**	**7**	**7**	**7**	**6**	**4**	**5**	**3**	**3**

6.4.3. Other Indicators

The implicit interest indicators *bookmark*, *save*, *print*, and *memo* had lower usage than the other indicators mentioned above. Users bookmarked or printed only a few web pages while surfing web. Users did not bookmark all interesting web pages, so if used alone they cannot be used to identify all of the pages that a user finds interesting. However, these indicators have a very high accuracy when they are used, and they can be used together with other more frequently used indicators.

The results for the *bookmark, save*, *print*, and *memo* indicators are listed in Table 24. The first column is the indicator, the second column is the score (1-"not interested", 3-"interested" and 5-"very interested"); the third column is the sum of the usages for the specified indicator across 11 volunteers. The rest of the columns are detailed usages for each user. The value in each cell is the number of times that the indicator was used. The number of times each indicator was used varied significantly between each individual. For instance, for some users the *bookmark* indicator was a clearer indicator than other ones – user 5; for some other users *save* was a clearer indicator – user 10.

Of the web pages that were bookmarked, 95% of them were scored more than or equal to "interested" (3). The sum of bookmarked web pages across 11 volunteers tells us that users rarely bookmarked uninteresting web pages – no bookmarked web pages were scored as "not interested". User 1 and 5 showed a tendency of book-marking more web pages as the web pages became more interesting. These results indicate that *bookmark* was a good indicator.

Saved web pages were scored more than or equal to "interested" 98% of the time. This means that users rarely saved uninteresting web pages. Saved web pages were never

scored as “not interested.” All users, except user 8, only saved pages that they found interesting. Users 3, 6, and 10 showed a tendency of saving more web pages as the web pages became more interesting. These results indicate that *save* is a good implicit indicator.

All of the printed web pages were scored more than or equal to “interested”. This result tells us that users did not print uninteresting web pages. User 2, 3, 6, and 10 showed a tendency of saving more web pages as the web pages were getting more interesting. These results indicate that *print* is a good indicator.

Nearly all (98%) of the memoed web pages were scored more than or equal to “interested.” No memoed web pages were scored as “not interested.” No user other than user 9 memoed on web pages for which he was less than “interested.” User 1 did not used the *memo*, but user 3, 5, and 10 showed a tendency of saving more memos as the web pages became more interesting. These results also indicate that *memo* is a good indicator.

Table 24. Results of *bookmark*, *save*, *print*, *memo* indicators

			Users										
Indicator	**Score**	**Sum**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
bookmark	1	**0**	0	0	0	0	0	0	0	0	0	0	0
	2	**5**	0	1	0	0	0	0	1	2	0	0	1
	3	**24**	2	6	1	2	1	0	2	5	0	2	3
	4	**31**	2	3	0	1	6	4	1	2	3	7	2
	5	**41**	5	7	6	1	9	1	3	1	2	6	0
save	1	**0**	0	0	0	0	0	0	0	0	0	0	0
	2	**1**	0	0	0	0	0	0	0	1	0	0	0
	3	**12**	0	8	1	0	0	1	0	2	0	0	0
	4	**15**	0	4	3	5	0	1	0	0	0	2	0
	5	**29**	0	10	6	0	1	3	0	1	2	6	0
print	1	**0**	0	0	0	0	0	0	0	0	0	0	0
	2	**0**	0	0	0	0	0	0	0	0	0	0	0
	3	**12**	0	2	0	1	0	1	5	2	0	0	1
	4	**11**	0	4	1	2	0	1	0	0	0	2	1
	5	**40**	0	15	7	1	3	4	2	1	4	3	0
memo	1	**0**	0	0	0	0	0	0	0	0	0	0	0
	2	**1**	0	0	0	0	0	0	0	0	1	0	0
	3	**8**	0	3	0	1	0	1	2	1	0	0	0
	4	**12**	0	1	2	2	1	0	0	0	3	1	2
	5	**30**	0	9	10	0	2	0	0	1	1	7	0

6.5. Summary

This paper identifies several implicit indicators that can be used to determine a user's interest in a web page. This paper evaluates both previously studied implicit indicators and several new implicit indicators. All indicators examined were *complete duration*, *active window duration*, *look at it duration*, *distance of mouse movement*, *number of mouse clicks*, *distance of scrollbar movement*, *number of scrollbar clicks*, *number of key up and down*, and *size of highlighting text*. The data was 11 users' implicit indicator data and a 1-5 interest rating of each page. During our experiment volunteers were encouraged to behave normally.

Two evaluation criteria were used: (1) how accurately an indicator can predict users' interests and (2) how many users' interests an indicator can predict. We used two data sets: "visits with maximum duration" and "all visits". We believe that "visits with maximum duration" is more useful for prediction than "all visits", because users did not tend to read a web page again, once users read about the web page (Billsus and Pazzani, 1999). Over the data set containing "visits with maximum duration", the implicit interest indicators *Complete*, *Active*, *LookAtIt*, and *MousMove* were able to predict 8 users' interests towards web pages, but over the data set of "all visits" the indicators were able to predict only 7 users' interests. These facts also notified that the "visits with maximum duration" data set is more useful in predicting users' interests more accurately than the data set of "all visits".

The experimental results told us that *MousMove* could be the most practical indicator because this event is simple to detect and has less risk than *Active*. If a user leaves a web page open and leaves the room, the *MousMove* indicator will not be affected. The

indicator of *MousClk#* was the next best indicator, which was recognized as the best in (Jung, 2001). Our results indicate that there was no indicator that was valid for all users. Depending on the user, an indicator may or may not be valid.

We also evaluated less-frequently-used indicators of user interest: *bookmark*, *save*, *print*, and *memo*. When we divided the data set less than "interested" and more than or equal to "interested", "95% of the bookmarked web pages, 98% of the saved web pages, 100% of the printed web pages, and 98% of the memoed web pages belonged to the score of more than or equal to "interested".

We expected that the *LookAtIt* indicator would be more accurate than the *Complete* and *Active* indicators, but the results for all three were similar. We believe that this was because volunteers did not move around much and looked at the monitor most of the time while browsing. Perhaps a longer evaluation would give more accurate results for the *LookAtIt* indicator, since users would act more naturally after more than 1 or 2 hours of surfing. We can combine this indicator to an application for personalized web search results in the future. The collected interesting web pages for a user can be used for building a user interest hierarchy.

Chapter 7

Related Work

The adaptive web is a relatively young research area, starting in early 1990. Now it attracts many researchers from different communities: machine learning, information retrieval, user modeling, and web-based education (Brusilovsky and Maybury, 2002). Our goal is to build user interest models implicitly and incorporate them to personalized web search. Thus, we review web information retrieval, user modeling, and machine learning. We discuss each of these categories in turn.

7.1. Web Information Retrieval

Web information retrieval (WIR) systems gather information from web pages or users who are using web pages. In this section we overview basic steps of a WIR. Furthermore, we overview those adaptive web systems that do not include personalized user modeling such as recommendation systems (collaborative filtering systems) that rely on the similarity between a user's preference and that of other people. The six sub sections are listed in Figure 34

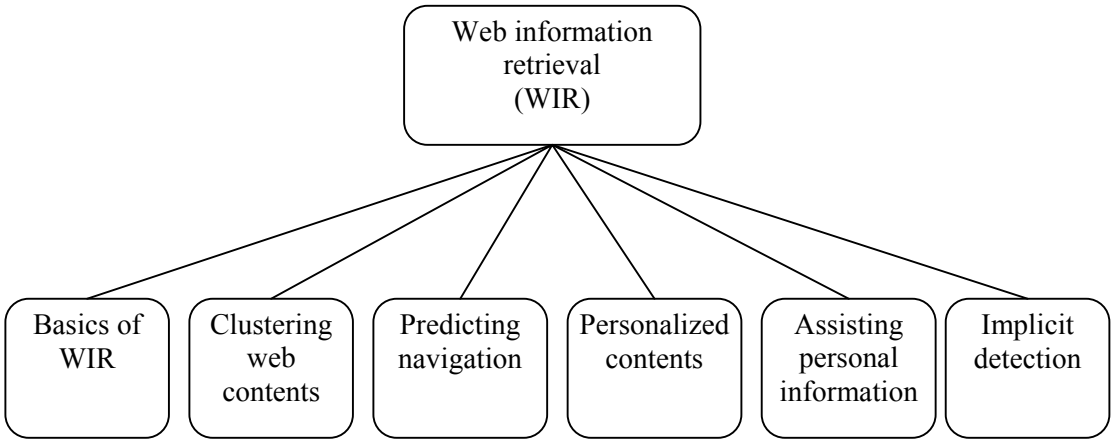


Figure 34. Diagram of web information retrieval

7.1.1. Basics of a WIR System

Many WIR systems use a model based on word frequency information to identify relevant documents (Zamir and Etzioni, 1999; Cutting et al., 1992). It is important to describe the process by which the computer converts text into a form that can be processed. For most WIR systems, the most basic unit of text analysis is the word, while phrases, sentences, or paragraph may be more meaningful.

7.1.1.1. Lexical Analysis

Text processing converts the text into a stream of tokens, including numbers, abbreviations, and alphanumeric sequences. There exists a large class of words - called stop list - that have no inherent meaning when taken out of context (e.g., “a”, “the”, “are”, or “to”). By removing the stop list from web documents, a web information retrieval system can significantly increase its efficiency like reducing the time and memory space required for running the system.

Opinion varies as to the optimal size of a stop list, although a larger size is preferred. The size and content may be domain dependent (Hull, 1994). The stop list should not be selected solely on the basis of frequency, because some frequent words still bear important semantic meaning in a document. In some researches (Croft, 1991; Zamir and Etzioni, 1998) both too frequent words and too rare words were removed. It has been found in retrieval experiments that using a stop list in the range of 8 through 500 words does not reduce the accuracy of search algorithms in identifying relevant documents (Frakes and Baeza-Yates, 1992).

Another common strategy for reducing text size and potentially improving WIR systems is to apply a stemming algorithm to word tokens. A stemming algorithm is a linguistic tool for building word equivalence classes by removing and modifying prefixes and suffixes to identify the root form of the word. This idea is based on the assumption that common morphological variants of a word have similar meanings. For example, a user who is interested in the word "computer" may also be interested in the word of "computing", "computerized", "compute" etc. Search engines are able to reduce the size of the index, often as much as 20-50% by applying a stemming method (Hull, 1994). Frakes and Baeza-Yates (1992) conducted a large number experiments to test the performance of stemming algorithm. In general, it appears that stemmers do not degrade retrieval performance, and the specific choice of stemmer does not seem to be important.

7.1.1.2. Phrase

Describing concepts often requires more than a single word. A composed term using two or more single words (called "phrase") usually has more specific meaning and can disambiguate related words. For instance, "apple" has different meanings in "apple tree" and in "apple computer". Using phrases, in addition to words, can improve the performance of WIR systems. Statistical phrase-finding algorithms are mainly used for improving the performance of information retrieval (Kim and Chan, 2003; Turpin and Moffat, 1999; Croft et al., 1991; Fagan, 1987). There are three main approaches: syntactic (Lima and Pedersen, 1999), statistical (Mitra et al., 1997), and hybridized (Gokcay and Gokcay, 1995). Our research mainly focuses on the statistical approach, which does not need any grammatical knowledge and has easy adaptability to other languages. Statistical

phrase-finding approaches have been used for expanding vector dimensions in clustering multiple documents (Turpin and Moffat, 1999; Wu and Gunopulos, 2002), or finding more descriptive or important/meaningful phrases (Ahonen et al., 1998; Chan, 1999).

Wu and Gunopulos (2002) examined the usefulness of phrases as terms in vector-based document classification. They used statistical techniques to extract phrases from documents whose document frequency (*df*) is larger than or at least equal to a predefined threshold. Fagan (1987) selected phrases having a document frequency of at least 55 and a high co-occurrence in the same sentence. Mitra et al. (1997) collected all pairs of non-function words that occur contiguously in at least 25 documents. Turpin and Moffat (1999) used Mitra's method for statistical phrases for vector-space retrieval. Since the aim of these approaches was to find the significant words or phrases among documents, this method could remove meaningful phrases in a document. These all methods focused on collecting two-word phrases.

Croft, et al. (1991) described an approach where phrases identified in natural language queries are used to build structured queries for probabilistic retrieval models and showed that using phrases could improve performance. They used *tf*×*idf* (term frequency inverse document frequency) information for a similarity measure. Croft (2000) segmented a document's text using a number of phrase separators such as verbs, numbers, dates, title words, format changes, etc. Next, his method checks the candidate phrases to see if they are syntactically correct. Finally, the occurrence frequency of the remaining phrases is checked.

Gokcay and Gokcay (1995) used statistically extracted keywords and phrases for title generation. Their statistical method used grammatical information of tags and

sentences, but it is hard to determine a sentence without grammatical information. They used the cosine correlation function for comparing the similarity of two words.

Ahonen's method (Ahonen et al., 1998) finds all possible combinations of words within a fixed window using Mannila and Toivonen's (1996) algorithm. Suppose the window size is 6 and the string in that window is "abcdef". Their algorithm generates all possible cases: "ab", "bc", "cd", "de", "ef", "abc", "bcd",… "bcdef", "abcdef". Then, it computes the conditional probability for the weight of those phrases. A phrase "abc" has two possible weights from *P*("c"| "ab") and *P*("bc"| "a"), from which the higher value is chosen. They also allowed gaps within a phrase. The use of two parameters – a threshold to remove less descriptive phrases in the generating stage and a threshold for the maximum phrase length – could be too strict.

Zamir's algorithm (Zamir and Etzioni, 1998) uses only frequency information. They collect neither too frequent nor too rare phrases. This method needs two user-defined parameters: one for removing too rare or too frequent words and the other for selecting phrases out of all possible phrases.

Chan's algorithm (1999) improved performance by using correlation information within a phrase. His algorithm calculates the correlation values of all pairs. The main drawback of this method resides in its incompleteness. Suppose there is a string *S*="abaxbxaxxbxaxxxb…xxbbxbxbxxbxxxb…", where 'a' and 'b' represent words, and 'x' represents any word. If all correlations between 'a' and 'b' with 1 through 4 distances, and correlations between 'b' and 'b' with 1 through 4 distances have values higher than the threshold, Chan's algorithm will generate a word "abbbb" that does not exist in the string. These cases are very unlikely to happen in a normal article such as newspaper or journal

article. But web pages contain lists of similar product names or tables that just arrange a few different repeating words many times. We experienced these non-existing words in our experiment such as “test pass test”, “student test pass”, “teach assist teach class”, etc. Another disadvantage of Chan’s algorithm is that it requires a user-defined maximum phrase length. Chan (1999) implemented the algorithm in time $O(n^{w2})$, where n^w is the number of distinct words, while our implementation consumes $O(n_w)$.

We compare previous statistical approaches and attempts to find meaningful phrases in a document. The length of the phrases is various like normal phrases and our algorithm requires no specified parameters. We mainly focus on a statistical approach without using syntactic information. Simple phrase separators (e.g., stop-words and non-alphabet characters) are used only. Our research experimentally shows which correlation functions are better than others in terms of measuring word correlation in our variable-length phrase-finding algorithm.

7.1.2. Clustering Web Contents

Methods about clustering web contents group related web pages together (e.g., Vivisimo (2005)). It does not use any personal information. These methods focus on clustering large amount of information in a short time.

Scatter/Gather (Cutting et al., 1992) supports a hierarchical interface by clustering documents into topically coherent groups and providing descriptive summaries. A user can select one or more basic clusters to focus on, recursively, in the subset of documents. The user specifies a query and two thresholds (the # of documents to be initially retrieved and

the # of sub-clusters). Scatter/Gather produces significant improvements over similarity search ranking.

STC (Suffix Tree Clustering) (Zamir and Etzioni, 1998) is a linear clustering algorithm in terms of the document size, which is based on the phrases (an ordered sequence of one or more words) that are common to groups of documents. STC has three steps: document cleaning, identifying base features using a suffix tree, and merging these base features into clusters.

Grouper (Zamir and Etzioni, 1999) is an interface that dynamically groups the search results into clusters labeled by phrases extracted from the snippets – this uses STC as its clustering algorithm. Given a set of phrases from STC, Grouper presents them in descending order of coverage (the percentage of documents in the cluster that contain the phrase) and length (# of words in the phrase, not counting stopped words). Grouper creates clusters by merging base features (a phrase and the set of documents that contain it). However, when the clusters fail to capture the semantic distinctions the users were expecting, it can be confusing.

Newsblaster (Barzilay et al., 1999) generates a concise summary by identifying and synthesizing similar elements across related text from a set of multiple documents. Common phrases are identified across sentences assuming a set of similar sentences as input extracted from multiple documents on the same event. Using language generation to merge similar information is a new approach that significantly improves the quality of the resulting summaries. This technique may be able to be used for summarizing web pages.

7.1.3. Predicting Navigation

Another approach of web personalization is to predict forward references based on partial knowledge about the history of a session. This approach predicts future requests and present documents to client. When the server guesses correctly, the latency of the next request is greatly reduced; and if the server guesses incorrectly, the client requests the intended next document. These techniques also use other people's information, and can support pre-fetching. The drawback is that it is difficult for them to predict previously unvisited pages.

The two major approaches for predicting navigation are Markov models and n-gram models. Zukerman et al. (1999) and Cadez et al. (2000) use a Markov model to learn and represent significant dependencies among page references. The n-gram models predict which URL will be requested next; the Markov models compute the probability of next request. An n-gram is a sequence of n web requests, and the n-gram models learn how often each sequence of n requests was made in the training data.

Deshpande and Karypis (2001) use pruning techniques to reduce the model size and improve predictions. Their method intelligently selects parts of different order Markov models so that the resulting model has a reduced state complexity and improved prediction accuracy.

WebCANVAS (Cadez et al., 2000) visualizes the similar group of users by using a Markov model – this is illustrated on user-traffic data from msnbc.com. Zukerman et al. (1999) developed a system that reduces a user's expected waiting time by pre-sending documents s/he is likely to request. Their models are based on observing the behavior patterns of many users, rather than using the behavior of an individual user. They combine

two features: the order in which documents are requested and the structure of the server site. Mladenić (2000) used naïve Bayes or k-nearest neighbor models to predict the next link.

SurfLen (Fu et al., 2000) actively monitors and tracks a user's navigation. Once a user's navigation history is captured, they discover the hidden knowledge contained in the history by applying association rule mining techniques. This uses a form of "market basket" analysis (Agrawal and Srikant, 1994). The knowledge is then used to recommend potentially interesting web pages to users. PageGather (Perkowitz and Etzioni, 2000) builds a co-occurrence matrix of all pairs of pages visited and finds clusters of pages that are frequently viewed in the same session. Like SurfLen, PageGather also recommends the top n pages that are most likely to co-occur with the visitor's current session.

Shahabi and Banaei-Kashani (2003) proposed a web-usage-mining framework using navigation pattern information. They introduced a feature-matrices model (FM) to discover and interpret users' access patterns.

This approach is different from ours since we do not use navigation pattern information but rather the contents of web pages. Both the n-gram and the Markov methods require large volumes of training data and cannot generate previously unvisited pages because they use navigation pattern information.

7.1.4. Personalized Contents

The methods related to personalized contents use user's access pattern to provide personalized web services. Server usually monitors the user's access patterns. This technique identifies a set of categories from other users' access pattern and then provides a

new user the closest category based on his/her access pattern. Our method does not personalize the set of categories, but personalizes results returned from a search engine.

Page et al. (1998) first proposed personalized web searches based on modifying the global PageRank algorithm with the input of bookmarks or homepages of a user. Their work mainly focuses on global "importance" by taking advantage of the link structure of the web. Haveliwala (1999) determined that PageRank could be computed for very large subgraphs of the web on machines with limited main memory. Brin et al. (1998) suggested the idea of biasing the PageRank computation for the purpose of personalization, but it was never fully explored. Haveliwala (2002) used personalized PageRank scores to enable "topic sensitive" web search. He concluded that the use of personalized PageRank scores can improve web search, but the number of hub vectors (e.g., number of interesting web pages used in a bookmark) used was limited to 16 due to the computational requirements. Jeh and Widom (2003) scaled the number of hub pages beyond 16 for finer-grained personalization.

Liu et al. (2002) also tried mapping user queries to sets of categories. This set of categories served as a context to disambiguate the words in the user's query, which is similar to Vivisimo (2005). They studied how to supply, for each user, a small set of categories as a context for each query submitted by the user, based on his or her search history.

Anderson (2002) proposed PROTEUS (This system personalizes web browsing for visitors using wireless PDAs at many web sites, adapting each site in turn), MINPATH (This algorithm finds personalized shortcut links efficiently), and MONTAGE (This system supports personalized, dynamic portals of web content, based on the navigation behavior of

each individual visitor), which personalize the web content for different audience, that personalize individual pages (e.g., elide-content), site-sessions (add-shortcut), or entire browsing sessions.

Footprints (Wexelblat and Maes, 1997) helps user browse complex web sites by visualizing the paths taken by users who have been to the site before. The paths are visualized as a graph of linked document nodes – color represents the frequency of use of the different paths. Footprints are left automatically by anonymous different users, and new visitors do not need to provide any information about themselves in order to use the system. However, users can only see the frequency of the links between adjacent pages.

Perkowitz and Etzioni (1997, 2000) find singular transformations that appeal to all visitors at the site by synthesizing index pages – hubs of links to other pages in the site. They consider the problem of index page synthesis and sketch a solution that relies on novel clustering and conceptual clustering techniques.

Yan et al. (1996) present a system that facilitates the analysis of past user access patterns to discover common user access behavior (for example, navigation through the men's clothing department, consumer electronics, and traveling). They perform clustering over a web site's logs. Once this information is analyzed, it is used to improve the static hypertext structure or to dynamically insert links to web pages. They model visitors as vectors in URL-space (an n-dimensional space with a separate dimension for each page at a site) and cluster them using "leader algorithm" (Hartigan, 1975).

7.1.5. Assisting Personal Information

Methods for assisting personal information help a user to organize their own information better and increase the web usability. The assistant usually resides in a user's personal computer. These techniques do not use other people's information and are not related to predicting navigation.

PowerBookmarks (Li et al., 1999) is a web information organization, sharing, and management tool, which monitors and utilizes users' access patterns to provide useful personalized services. PowerBookmarks provides automated URL bookmarking, document refreshing, bookmark expiration, and subscription services for new or updated documents. BookmarkOrganizer (Maarek and Ben-Shaul, 1996) is an automated system that maintains a hierarchical organization of a user's bookmarks using the classical HAC algorithm (Voorhees, 1986), but by applying "slicing" technique (slice the tree at regular intervals and collapse into one single level all levels between two slices). Both BookmarkOrganizer and PowerBookmarks reduce the effort required to maintain the bookmark, but they are insensitive to the context browsed by users and do not have reordering functions.

7.1.6. Implicit Detection of User's Characteristics

Detecting the interests of a web page from a person can happen in either Client or Server. Obtaining labeled training instances is necessary for agents to learn a user's interest; however, how the learning algorithm obtains training examples is an important issue.

Jung (2001) developed Kixbrowser, a custom web browser that recorded users' explicit rating for web pages and their actions: *mouse clicks*, *highlight*, *key input*, *size*, *copy*,

rollover, *mouse movement*, *add to bookmark*, *select all*, *page source*, *print*, *forward*, *stop*, *duration*, the *number of visits* (frequency), and *recency* during users' browsing. He developed individual linear and nonlinear regression models to predict the explicit rating. His results indicate that the *number of mouse clicks* is the most accurate indicator for predicting a user's interest level.

CuriousBrowser (Claypool et al., 2001) is a web browser that recorded the actions (implicit ratings) and explicit ratings of users. This browser was used to record *mouse clicks*, *mouse movement*, *scrolling* and *elapsed time*. The results indicate that the time spent on a page, the amount of *scrolling* on a page, and the combination of time and *scrolling* has a strong correlation with explicit interest.

The two experiments above show some inconsistency. Jung (2001) said *mouse click* is a good indicator, but Claypool et al. (2001) did not. Jung (2001) found that *duration* and *scrollbar movement* are not very predictive of a user's interest, but Claypool et al. (2001) said they are good indicators.

Powerize (Kim et al., 2001) is a content-based information filtering and retrieval system that uses an explicit user interest model. They also reported a way to implement the implicit feedback technique of user modelling for Powerize. They also found that observing the printing of web pages along with reading time could increase the prediction rate for detecting relevant documents.

Goecks and Shavlik (2000) proposed an approach for an intelligent web browser that is able to learn a user's interest without the need for explicitly rating pages. They measured *mouse movement* and *scrolling* activity in addition to user browsing activity (e.g., navigation history). We extend these existing implicit interest indicators in this research.

Granka et al. (2004) measured eye-tracking to determine how the displayed web pages are actually viewed. Their experimental environment was restricted to a search results.

We examine the duration implicit indicator in more detail. We divide the duration into three types: *complete duration*, *active window duration*, and *look at it duration*. Our *complete duration* is different from the duration in Jung's (2001) work. His duration includes the downloading time of a web page, but ours does not. We divided the web pages visited during our evaluation into two groups: (1) web pages that a user visited more than once and viewed for the longest duration, and (2) all web pages that were visited more than once, while Jung (2001) only used the second data set. In our experiment, we let a user navigate to any web page and do normal tasks such as using chat programs or word processors during the experiment. Another difference is that we use head orientation instead of eye-tracking (Granka et al., 2004). Our experiment is also valuable since there are cases where an application does not have devices for tracking a user's eyes.

7.2. User Modeling

This section lists adaptive systems that use user modeling. The primary goal of user modeling is to enable the prediction of a user's actions on a personalized web site, and thus to help determine which adaptation are useful for the user and navigate the web. The forms of user model are as varied as the purposes for which user models are formed as shown in Figure 35. Mainly user models try to describe (Webb et al., 2001): the cognitive processes of user's action, the difference between the user's skill and expert skills, the user's behavioral pattern or preferences, and the user's characteristics. Another important

dimension is to distinguish whether models are based on individual users or communities of users (Webb et al., 2001). Whereas much of the academic research is related to modeling individual users, many applications (Ungar and Foster, 1998) in electronic commerce are related to forming generic models of user communities.

User modeling poses a number of challenges for machine learning, including: computational complexity, concept drift, the need for labeled data, and the need for large data sets. User modeling is known to be a very dynamic modeling task – attributes are changing over time. The capability of adjusting to these changes quickly is known as "concept drift" (Widmer and Kubat, 1996). Webb et al. (2001) examined each of these issues and reviewed approaches. These techniques can reside on both client/server sides. Generally these techniques do not use other people's information. These can support prefetching and advise unvisited pages.

7.2.1. Adaptive Hypermedia

Adaptive hypermedia focuses on improving web (Hypermedia) interactions by modeling users and adapting the experience. The differences from adaptive web sites lies in the application domain – Hypermedia is related to help systems (adapting to the particular context of the help request), information retrieval (helping users find as much relevant content as possible), or online information systems (helping users find high-quality content quickly). Brusilovsky (2001) introduced this field for newcomers by an overview. Previous empirical studies have shown that adaptive navigation support can improve the speed of navigation (Kaplan et al., 1993) and learning (Brusilovsky and Pesin,

1998). The adaptive presentation can also affect the understanding of content (Boyle and Encarnacion, 1994).

Weber and Specht (1997) demonstrated that user modeling techniques like simple overlay models or more elaborated episodic learner models are effective for adaptive guidance and for individualized help in web-based learning systems. This system uses a combination of an overlay model (provide default path and short cut path) and an episodic user model (stores knowledge about the learner in terms of a collection of episodes, such episodes can be viewed as cases). This system also supports adaptive navigation as individualized diagnosis and helps on problem solving tasks.

7.2.2. Human Behavior Based User Model

Human behavior based user models are not good at predicting unvisited web pages, because this approach utilizes models that are based upon user actions such as path, click, downloads, frequency of visits to a web page, etc.

Mobasher et al. (1999) proposed an approach to usage-based web personalization that takes into account both the offline tasks related to the mining of usage data and the online process of automatic web page customization. Their technique captures common user profiles based on association-rule discovery and usage-based clustering.

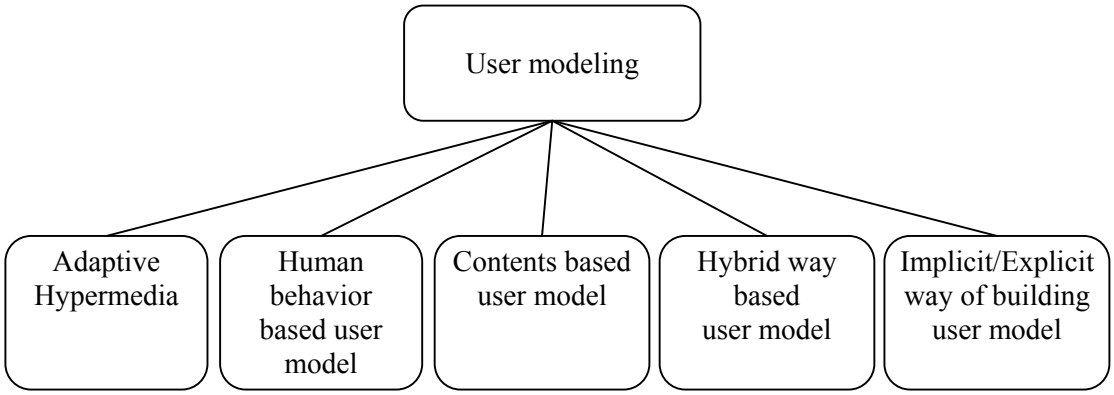


Figure 35. Diagram of user modeling

Letizia (Lieberman, 1995) is a user interface agent (client-side), which operates with conventional web browser. The agent tracks the user's browsing behavior (e.g., following links, initiating searches, and requests for help) and tries to estimate the user's interest in as-yet-unseen pages. Letizia can recommend nearby pages by doing lookahead search. Letizia cannot take advantage of the past experiences of other visitors to the same site, since it runs on a client.

Pazzani and Billsus (1997, 1999a, 1999b) state that a web site should be augmented with an intelligent agent to help visitors navigate the site, and should learn from the visitors to the web site. An agent can learn common access patterns of the site both by analyzing web logs and by inferring the visitor's interests from actions of the visitor.

TELLIM (Hoerding, 1999) monitors the behavior of a customer and recognizes the user's needs and preferences. This information adapts the product presentations. Using a set of rules, the system evaluates for every presentation element whether the customer was interested in it or not. Those rules are extracted from their personal experience. For example, if the downloading of an integrated image was interrupted, then it has negative interest to the customer. The attribute of an element is quite simple: kind of product (e.g., "car", brand, e.g. "Ford") and kind of information (e.g., "engine"). They used CDL4 algorithm to learn the preferences of the customer. The user model for each customer is expressed as a set of rules like if the size of the item is less than 20GB (hard drive), then the customer may not be interested. They addressed the refinement of the user model as a future work.

The AVANTI Project (Fink et al., 1996) focuses on helping users by adapting the content and the presentation of web pages to each individual user. The elderly and

handicapped users are also partly considered. AVANTI also relies partly on explicit profiles. It uses both user's path and his/her model to guess pages.

7.2.3. Contents Based User Model

Contents based user models can predict web pages unvisited by users. This is achieved because this model learns from the contents of web pages that a user visited. This technique usually has higher dimensional vectors.

WebWatcher (Joachims et al., 1997) is a tour guide software agent. It accompanies users from page to page providing several types of assistance: highlighted interesting hyperlinks, menu bar, and advice. It also learns from experience to improve its advice-giving skills. Since it runs as a centralized server it can leverage data from different users. User interest (user model) is represented by high-dimensional feature vectors, each dimension representing a word. This uses reinforcement learning (Sutton and Barto, 1998) to allow agents to learn control strategies that select optimal actions in certain settings.

SiteIF (Stefani and Strapparava, 1999) is a personal agent that follows users as they browse a web site. It learns user's interests from the requested pages and builds/updates a user model. This system builds the user model in the form of a semantic net whose nodes are concepts and arcs are the co-occurrence relation of two concepts. The relevance between user model and a document is estimated using the Semantic Network Value Technique.

Mobasher et al (1999) propose an approach to usage-based web personalization taking into account both the offline tasks related to mining of usage data and the online process of automatic web page customization. Their technique captures common user

profiles based on association-rule discovery and usage-based clustering. The advantage of this approach is that it can predict visited web pages well, but is not good for predicting unvisited web pages. Content-based user models are generated from the contents of web pages that a user has visited. This technique usually has higher dimensional vectors and needs a greater number of training data. The advantage is that it can predict unvisited web pages by users.

Syskill & Webert (Pazzani et al., 1996) is an intelligent agent that learns user profiles. After identifying informative words from web pages to use as Boolean features, it learns a Naïve Bayesian classifier to determine the interest of a page to a user. It converts the HTML source of a web page into a Boolean feature vector that indicates whether a particular word is present or absent in a particular web page. Hybrid models are learned by observing user's actions and the contents of web pages visited by a user.

Mobasher et al. (2000) combine site usage-based clustering and a site content-based approach to obtain uniform representation, in which the user preference is automatically learned from web usage data and integrated with domain knowledge and the site content. These profiles could be used to perform real-time personalization. Their experimental results indicate that the integration of usage and content mining increases the usefulness and accuracy of the resulting recommendations.

A news-agent called News Dude (Billsus and Pazzani, 1999), learns which stories in the news a user is interested in. The news-agent uses a multi-strategy machine learning approach to create separate models of a user's short-term and long-term interests. They use the Nearest Neighbor algorithm for modeling short-term interests and a Naïve Bayesian classifier for long-term interests.

Unlike News Dude that creates a model of two layers, our approach tries to model a continuum that spans from general to specific interests. Once we get a user profile based on contents, we can extend it to incorporate human behavior based user model.

7.2.4. Hybrid Way Based User Model

Hybrid approach uses both user's actions and the contents of the web pages visited by a user for building a user model. Mobasher et al. (2000) combine site usage based clustering and site contents based approach to obtain a uniform representation in which the user preference is automatically learned from web usage data and integrated with domain knowledge and the site contents. These profiles can be used to perform real-time personalization. Their experimental results indicate that the integration of usage and content mining increases the usefulness and accuracy of the resulting recommendations.

7.2.5. Explicit/Implicit Way of Building a User Model

Most current approaches to personalization rely heavily on human participation to collect profile information about a user. The most common and obvious solution for collecting profile information about a user is asking for the user to specify their interests explicitly (Yahoo mail, 2003). However, the explicit approach has several disadvantages. Time and effort are required to specify interests, and user's interests may change over time. Alternatively, an implicit approach can identify a user's interests by inference.

Ardissono et al. (1999) demonstrated how user modeling and adaptive hypermedia techniques could be applied to present the most appropriate set of news (and advertisement) to each user. This system builds the initial model of a new user by asking

questions directly such as age, gender, job, hobbies, etc. Since the initial stereotype user model may be not accurate, the model is refined periodically after monitoring the user's behavior (e.g., which news s/he selects). The obtained user models are used for dynamic generation of the web pages based on a knowledge base (e.g., which news, at which detail level and which advertisement). However, setting rules for revising user profile and for predicting probability is difficult.

7.3. Machine Learning

Machine learning has two different methods for leaning: supervised learning and unsupervised leaning. Supervised leaning has labels on learning data sets, but unsupervised learning does not. Supervised learning can be divided into two sub categories: characterization and classification. Characterization methods learn from a set of good data, and then detect no-goods based on the learning. This technique is mainly used for anomaly detection. Classification accepts a set of labeled data and then learns from them. Unsupervised learning has two sub categories: clustering and outlier detection. Both methods use unlabeled data. Clustering tries to group similar elements. Outlier detection also groups similar elements, but rejects far elements at the same time. Outlier detection is commonly used for anomaly detection. Theses categories can be depicted as shown in Figure 36.

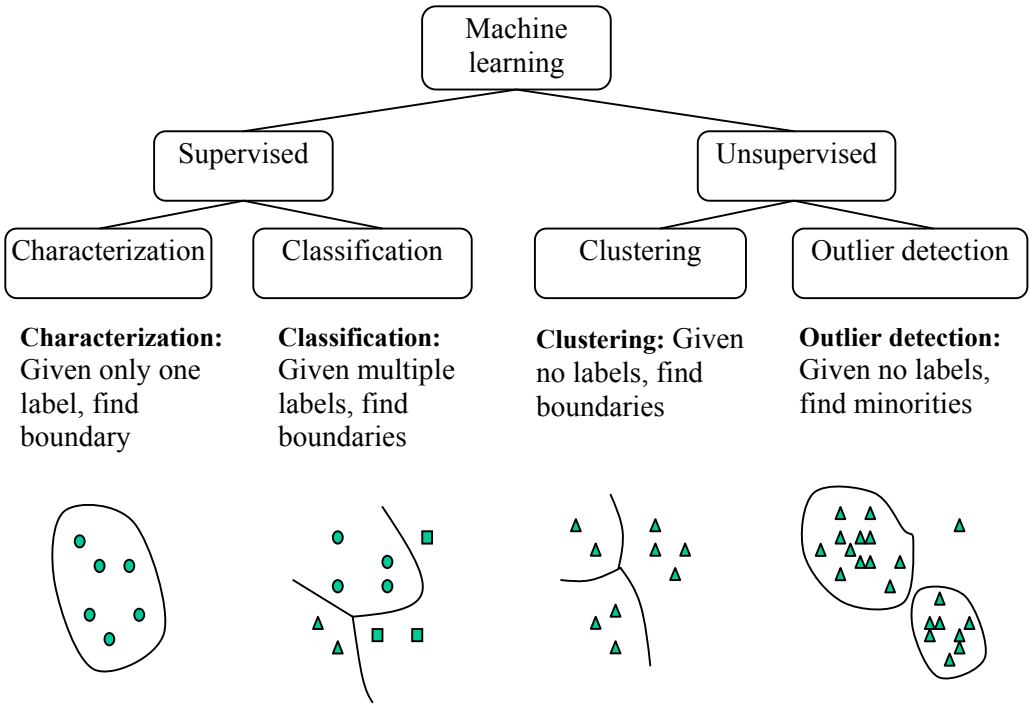


Figure 36. Diagram of machine learning

Classification can also be sub divided into symbolic and numerical methods as shown in Figure 37. Generally numerical method is hard for humans to understand in terms of how it is educated; but human can easily interpret the learned results by symbolic method. Following study mainly focuses on symbolic method, numeric method, and clustering techniques.

Some web personalization systems use machine learning to model user interest model or visitor's behavior. In their work, they have a set of web pages, which have category labels (e.g. interesting, uninteresting, or topics of interest). The task is to assign labels to the unseen web pages.

7.3.1. Symbolic Methods of Learning

Symbolic methods of learning yield results that can be readily understood by users. This ease of user interface contributes to understanding the learning methods and estimating their performance.

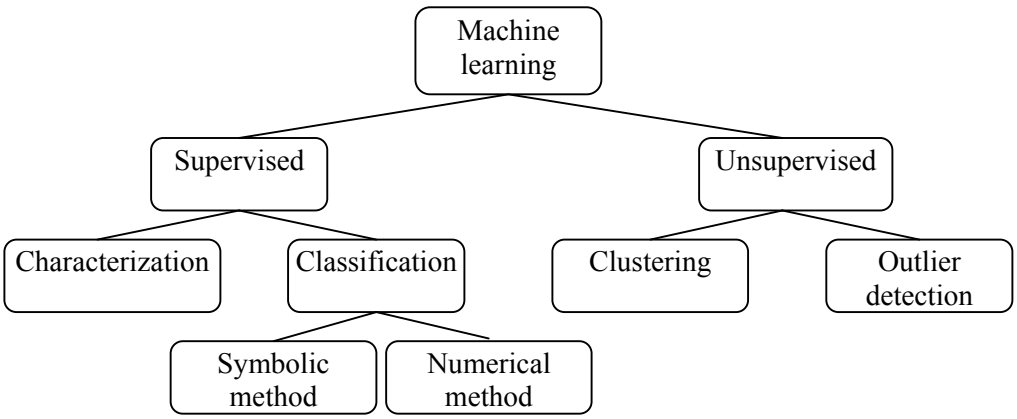


Figure 37. Diagram of classification

7.3.1.1. Semantic Networks

Semantic networks use objects as nodes in a graph, where the nodes are organized in a taxonomic structure and arrows represent relations between nodes. Each node represents a separate concept and links between nodes have several types of semantic relations. SiteIF (Stefani and Strapparava, 1999) represents a user model as a semantic network, whose nodes are concepts and arcs between nodes are the co-occurrence relation of two concepts. These methods learn the relationship between two concepts. It is an effective way to represent data as they incorporate the inheritance mechanism that prevents duplication of data.

7.3.1.2. Learning Decision Trees

Decision tree, such as ID3, ASSISTANT, and C4.5, generates a nested set of if-then-else rules, which is of the form, "if attribute < value then …". These algorithms recursively construct a tree using a greedy algorithm by finding the next attribute that maximizes the separation of categories at each node. Their inductive bias is a preference for small trees over large trees, Occam's razor.

7.3.1.3. Learning Sets of Rules

Rule-learning algorithms learn rule sets to assign categories to a training set. RIPPER (Cohen, 1995) is a rule learning algorithms that perform efficiently on large noisy datasets. It uses greedy algorithm, so the early rules generated cover more number of training instances. Apriori (Agrawal and Srikant, 1994) generates all significant association rules between the categories and other attributes in large databases. Given a training set, it

allows any attribute to serve as the label and finds if-then rules among attributes. LERAD (Mahoney and Chan, 2003) also finds association rules by choosing a set of rules randomly and applying rule validation.

Genetic algorithms encode each rule set as a bit string and use genetic search operators to explore this hypothesis space. It operates by iteratively updating a pool of rules, called population. On each generation, only some of the rules are selected according to their fitness and are carried forward into the next generation's population intact.

SurfLen (Fu et al., 2000) uses rules in recommending next web pages. TELLIM (Hoerding, 1999) represents user profile as a set of rules. TELLIM handles only some product items, and measures if a customer is interested in the item or not. Their rules consist of not many precedents and antecedents. However in our study there are many attributes (all distinct words in web pages), which can reduce the speed of the rule-learning algorithm.

7.3.2. Numerical Methods of Learning

If a symbolic method of learning can be called a white box method, a numerical method of learning will be a black box method. Black box methods do not provide human readable results. However, the performance is quite good such as with Neural Networks which is one of the most popular learning algorithms.

7.3.2.1. Hidden Markov Models

Hidden markov models (HMM) are to model sequences of data. It can be viewed as a stochastic generalization of finite-state automata, where both the transitions between

states and the generation of output symbols are governed by probability distributions. Deshpande and Karypis (2001) applied pruning techniques to HMM model in order to reduce the model size and improve the prediction. Anderson (2002) also used Markov models to provide a probability distribution over out-links. Like wise, this approach is mainly used in predicting navigation in a web server.

7.3.2.2. Naïve Bayes Classifier

Naïve Bayes classifier is a Bayesian learning method. It is called "naïve" because the attribute values are assumed to be conditionally independent. Even when this assumption is not met, the naïve Bayes classifier is often quite effective. Bayesian belief networks represent the sets of conditional independence assumptions among subsets of the attributes more effectively. Syskill & Webert (Pazzani et al., 1996) uses Naïve Bayes classifier in building user profiles. As a disadvantage, the interpretation of the results can be difficult since every vector has only probability values.

7.3.2.3. Artificial Neural Networks

Artificial neural networks is among the most effective learning methods currently known. For example BACKPROPAGATION algorithm has been surprisingly successful in many practical problems. Each vector element is assigned to an input neuron; each category to an output neuron. These neurons and other intermediate neurons are all connected by weights. The network is trained by incrementally adjusting the weights to correctly categorize the training set. Nonlinear functions of the input can be learned by

adding the intermediate neurons. Even though this approach shows good performance in various areas, it supports very little human interpretation.

7.3.2.4. Instance-based Learning

Instance-based learning such as nearest neighbor and locally weighted regression simply memorize the presented training data. When a new query instance is encountered, a set of similar instances is retrieved from memory and used to classify the new query instance. These methods can use more complex, symbolic representations for instances. The disadvantage of this approach is that the cost of classifying new instances can be high. For some application domains, it may be necessary to classify hundreds or thousands of web documents in a few seconds.

7.3.3. Clustering Techniques

Many web personalization systems use clustering techniques in building a user model. In our work, we try to group words according to their correlations. For example all class names are supposed to be in one cluster or some related words such as computer and monitor are also in the same cluster. There are five categories of major clustering methods (Han and Kamber, 2000) as shown in Figure 38.

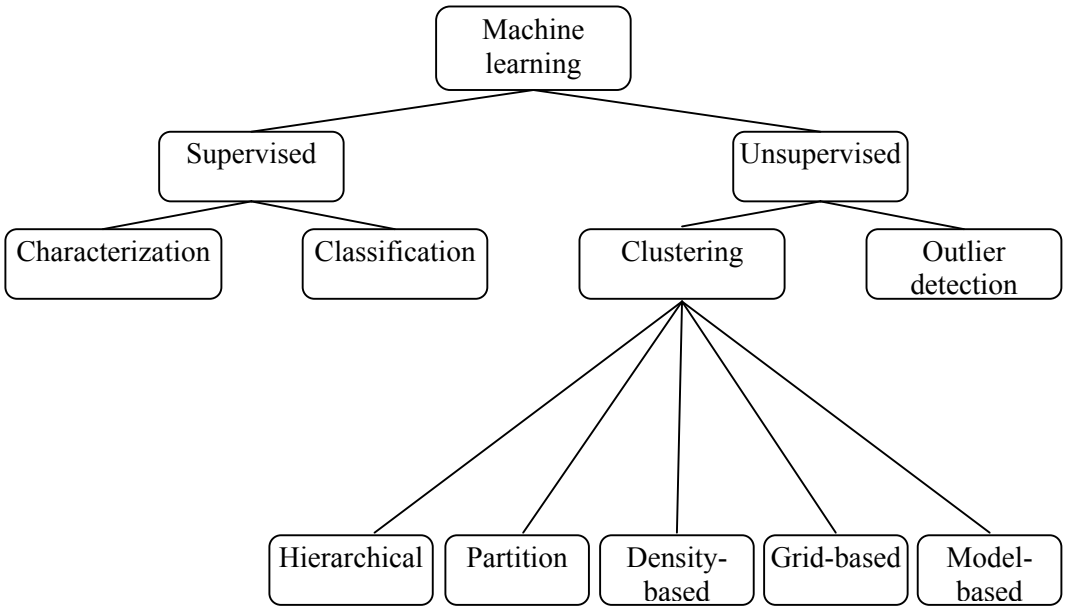


Figure 38. Diagram of clustering

Agglomerative (bottom-up) hierarchical clustering (AHC) algorithms initially put every object in its own cluster and then repeatedly merge similar clusters together, resulting in a tree shape structure that contains clustering information on many different levels (Voorhees, 1986). Merges are usually binary – merging two entities, which could be clusters or initial data points. Hence, each parent is forced to have two children in the hierarchy. Divisive (top-down) hierarchical clustering (DHC) algorithms are similar to agglomerative ones, except that initially all objects start in one cluster which is repeatedly split. These algorithms find the two furthest points, which are the two initial clusters. Then, the rest of the points are assigned to those two clusters depending on which one is closer. Hence, a binary tree is generated. Our DHC algorithm can generate multiple branches from one node depending on the data, which is the advantage of using a graph-partitioning technique. Several stopping criteria for AHC/DHC algorithms have been suggested, but they are typically predetermined constants – one common stopping criterion is the desired number of clusters (Fisher, 1987; Milligan and Cooper, 1985). These algorithms are very sensitive to the stopping criterion. The web documents, however, could be extremely varied (in the number, length, type and relevance of the terms/documents). When these algorithms mistakenly merge multiple "good" clusters due to the predetermined constraint, the resulting cluster could be meaningless to the user (Zamir and Etzioni, 1998). Another characteristic of the terms in web documents is that there reside many outliers. These outliers (sort of "noise") reduce the effectiveness of commonly used stopping criteria. AGNES (AGglomerative NESting) (Kaufman and Rousseeuw, 1990), DIANA(DIvisive ANAlysis) (Kaufman and Rousseeuw, 1990), BIRCH (Balanced Iterative Reducing and

Clustering using Hierarchies) (Zhang et al., 1996), CURE(Clustering Using REpresentatives) (Guha et al., 1998), ROCK (Guha et al, 1999), CHAMELEON (Karypis et al., 1999) all use these methods. In some application this binary split can be a major disadvantage. The hierarchical tree reorganized by slicing technique (Maarek and Ben-Shaul, 1996) can be useful in many areas.

Partitioning clustering algorithms such as the K-means algorithm initially creates a partitioning of K clusters. Those initial K clusters are then iteratively refined to achieve the final clustering of K clusters. A major drawback of this approach is that the number of clusters must be specified beforehand as an input parameter. Our algorithm only needs to cluster strongly connected words, but the K-means algorithm divides all words into K clusters without removing weak relations. We define the "strongly connected" in this paper as the relation between words whose correlation value is higher than the threshold. K-mean algorithm is too sensitive to outliers since an object with an extremely large value may substantially distort the distribution of data. Instead of taking the mean value, the medoid can be used, which is the most centrally located object in a cluster. PAM(Partitioning around Medoids) (Kaufman and Rousseeuw, 1990), CLARA(Clustering LARge Applications) (Kaufman and Rousseeuw, 1990), and CLARANS (Clustering Large Applications based upon RANdomized Search) (NG and Han, 1994) all use k-medoid method. We will not use this method in our work, because it is too difficult to find the best number of clusters initially.

Density-based clustering methods handles clusters with arbitrary shapes. This method regard clusters as dense regions of objects in the data space that are separated by regions of low density. Some examples of density-based approach are DBSCAN (Density

Based Spatial Clustering of Applications with Noise) (Ester et al., 1996), OPTICS (Ordering Points To Identify the Clustering) (Ankerst et al., 1999), and DENCLUE (DENsity-based CLUstEring) (Hinneburg and Keim, 1998).

Grid-based method uses a multi-resolution grid data structure. It quantizes the space into a finite number of smaller cells. On which all of the operations for clustering are performed. It is typically fast; the processing time is typically dependent on the number of cells not the number of data objects. Examples of grid-based methods are STING (STatistical INformation Grid approach) (Wang et al., 1997) and WaveCluster (Wang et al., 1997).

Model-based clustering methods set some mathematical model and then attempt to optimize the fit between the given data and the mathematical model. Such methods are based on the assumption that there is a mixture of underlying probability distributions that generate the data. COBWEB is an incremental conceptual clustering algorithm. Each cluster records the probability of each attribute and value, and the probabilities are updated every time an object is added to a cluster (Fisher, 1987). Instead of recalculating the whole probabilities of clusters to determine if child clusters are generated, our DHC algorithm uses a graph-based method and a different correlation function. CLASSIT (Gennari et al., 1989) is an extension of COBWEB for incremental clustering of continuous data set. It stores a mean or distribution for each individual attribute in each node and uses a modified similarity/dissimilarity measure that is an integral over continuous attributes. AutoClass (Cheeseman and Stutz, 1996) allows probabilistic membership of objects in clusters and hence clusters can overlap (an object belonging to multiple clusters). Also, it is a flat (non-hierarchical) clustering algorithm.

7.3.4. Correlation Functions

Correlation functions are used to measure the similarity/distance between objects within a clustering function or other machine learning algorithms.

Tan et al. (2002) demonstrated that not all functions are equally good at capturing the dependencies among variables and there is no single function that is consistently better than the others in all applications. They compared various existing correlation functions based on the key properties that Piatetsky-Shapiro (1991) presented and other properties of a correlation function. We propose some additional desirable properties for correlation functions.

Huang et al. (2002) applied a number of correlation functions in their application to identify interesting rules and sequential patterns from the Livelink log files. They presented a comparison of these measures based on the feedback from domain experts. Some of the interestingness measures were found to be better than others. Their experiment also supported that no single function is consistently better than the others. However, they did not analyze the desirable properties of correlation functions.

Piatetsky-Shapiro (1991) opened a new research area of information retrieval by presenting three desirable properties of a correlation function. However, his properties depend on *cross product ratio* (*CPR*) (Rosenfeld, 1994), in that all correlation function those satisfy CPR satisfies Piatetsky-Shapiro's properties. We propose two other desirable properties, which independent from *CPR*.

Jaroszewicz and Simovici (2004) presented a method of computing interestingness of item sets and attribute sets with respect to background knowledge encoded as a

Bayesian network. Their algorithm found interesting, unexpected patterns. Their work can be expanded to calculating all possible interestingness such as correlation or support between variables. Their work used the interestingness of dependency but did not focus on the property of correlation functions.

Chapter 8

Conclusions

We described a new approach of web personalization system and implemented this system. This consisted of 5 areas. First, to create a context for personalization, we proposed to establish a user interest hierarchy (UIH) that can represent a continuum of general to specific interests from a set of web pages interesting to a user. This approach was non-intrusive and allowed web pages to be associated with multiple clusters/topics. We evaluated the learned UIH based on data obtained from 13 users on our web server. Second, variable length phrase-finding algorithm found meaningful phrases. In order to choose the best method, we compared the number of matching phrases chosen by a method to those phrases chosen by 10 human subjects. Matched-pair design (Robertson, 1981), comparing the top 10 best measures, was used for evaluation. Third, we proposed two properties as desirable properties for correlation functions. In order to compare our 2 new desirable properties and previous 3 desirable properties, we collected 32 correlation functions and examined which correlation function satisfied which desirable properties. Fourth, we devised a new method of ranking web search results to serve each individual user's interests using UIH. We evaluated methods based on how many *interesting* web pages or *potentially interesting* web pages each algorithm found within certain number of top links (Bharat and Mihaila, 2001). Traditional precision/recall graphs (van Rijsbergen, 1979) were also used for evaluation. We counted which search term showed higher performances with our weighted scoring method (WS) than with Google as well. Fifth, this we identified several implicit indicators that can be used to determine a user's interest in a

web page. This paper evaluates both previously studied implicit indicators and several new implicit indicators. All indicators examined were *complete duration*, *active window duration*, *look at it duration*, *distance of mouse movement*, *number of mouse clicks*, *distance of scrollbar movement*, *number of scrollbar clicks*, *number of key up and down*, and *size of highlighting text*. The data was 11 users' implicit indicator data and a 1-5 interest rating of each page. During our experiment volunteers were encouraged to behave normally. Two evaluation criteria were used: (1) how accurately an indicator can predict users' interests and (2) how many users' interests an indicator can predict.

8.1. Summary of Contributions

The following is a summary of our contributions.

In Chapter 2, Learning Implicit User Interest Hierarchy for Context in Personalization, we represented user interest hierarchy (UIH) at different abstraction levels (general to specific), which could be learned implicitly from the contents (words/phrases) in a set of web pages bookmarked by a user. The higher-level interests were more general, while the lower-level ones were more specific. In order to build a UIH, we devised a divisive graph-based hierarchical clustering algorithm (DHC), which constructed a UIH by grouping words (topics) into a hierarchy instead of flat cluster used by STC (Zamir and Etzioni, 1998). Between the root and the leaves, "internal" tree nodes represented different levels of generality and duration of interest. Towards the root of a UIH, more general (passive) interests were represented by larger clusters of words while towards the leaves, more specific (active) interests were represented by smaller clusters of words. DHC automatically found the threshold for clusters of terms (words and phrases) where as STC

needed to specify the threshold. Furthermore, we used a more sophisticated correlation function, AEMI, than STC's conditional probability. We also observed that DHC with an AEMI correlation function and MaxChildren threshold-finding method made a more meaningful UIH with 59% of meaningful clusters than the other combinations. We added phrases to the words as feature. Our experimental results indicated that 64% of UIH were interpretable by human.

In Chapter 3, Identifying Variable-Length Meaningful Phrases with Correlation Functions, we proposed a variable-length phrase-finding algorithm (VPF), which found more meaningful phrases – VPF – than older methods – Ahonen's algorithm and Chan's algorithms. They regenerated sequences recursively with the words selected in the previous stage and searched for increased length of phrases in time $O(n_w)$, where n_w was the page size. This algorithm does not need any user-specified parameters, since our algorithm used average as a threshold and stopped when the length of phrases did not increase. The algorithm achieved further improved performance by pruning less meaningful phrases. The With-pruning was statistically significantly better than the Without-pruning with 95% confidence interval (P=0.004) for the VPF with F25. More meaningful phrases than previous methods were found by VPF with F25 and the improvement in performance is statistically significant than Ahonen's original method. We suspected the filtering stage of Ahonen's algorithm filtered many meaningful phrases out or their weighting scheme using the length of a phrase and tightness (Ahonen et al., 1998) distracted the correlation value of a phrase.

In Chapter 4, Analysis of Desirable Properties of Correlation Functions between Two Events, we identified 2 new desirable properties for a correlation functions in general. They were:

- if $P(A,B)$ increases when $P(A)$ and $P(B)$ remain unchanged, then F monotonically decreases;
- if $P(A)$ (or $P(B)$) and $P(A,B)$ increase at the same ratio, then F monotonically increases.

In addition, we revise and improve the first desirable property proposed by Piatetsky-Shapiro (1991) to make it more accurately descriptive. Some functions (e.g., *Odds*, *Conv*, *Inte*, and *Coll* in Appendix 1) produced a score of 1, even when two events were statistically independent.

- F can distinguish statistically independence of A and B.

Our empirical results indicated that our two new desirable properties were more descriptive than the previous three desirable properties provided in Piatetsky-Shapiro (1991). It was because all correlation functions that satisfy P1 also satisfied P2 and P3, and all correlation functions that satisfied P6 also satisfied P1-P3; but our properties were independent from P1 and P6. We tested our 2 new desirable properties over many different correlation functions and summarized their results with respect to each property. The summarized results included P1-P5, P6 that measured negative/positive correlation, and P7 that checked which correlation function returned a normalized return value. The associated table will help user compare the characteristics of correlations functions.

In Chapter 5, Personalized Ranking of Search Results with Implicitly Learned User Interest Hierarchies, we compared four ranking methods: Google, Random, US, and WS. We used two data sets: *interesting* web pages that are relevant to the user search term and *potentially interesting* web pages that could be relevant in the future. We introduced two personalized ranking methods (WS and US) that utilize an *implicitly* learned user profile (UIH). We built different UIHs for users depending on their interests. We identified four characteristics for terms that match the user profile and provide a probabilistic measure for each characteristic. The four characteristics were the level of a node where a term belongs to (*D*), the length of a term (*L*), the frequency of a term (*F*), and the emphasis of a term (*E*). *D* and *L* were calculated while building a UIH from the web pages in a user's bookmark. Different web pages had different values for *F* and *E* characteristics. Our experimental results indicate that WS method could achieve higher precision than Google for Top 10, 15 and 20 web pages that are relevant to the user search query. On *interesting* web pages, the Top link analysis showed WS achieved at least 13% higher precision than Google for Top 10, 15 and 20 links on average. WS outperformed US and Random in general also. On *potentially interesting* web pages, WS achieved the highest performance in all five Top links (Top 1, 5, 10, 15, and 20) as well. When incorporating the (*public*) ranking from the search engine, we found that equal weights for the public and personalized ranking can result in higher precision. A weight of 0.5 for the personal ranking seemed to show the highest performance on both data sets.

The precision/recall analysis visualizes the performance of each method in graphs. WS and US are closer to the upper-right corner than Google except with recall values lower than .15 (after Top 5). In general, WS outperforms US and Random for *interesting*

web pages. The results from precision/recall graph for *potentially interesting* web pages are similar. WS was closer to the upper-right corner than Google, US, and Random over all. These results conclude that WS could provide more accurate ranking than Google on average.

In Chapter 6, Implicit Indicators for Interesting web Pages, our experiments indicate that *complete duration*, *active window duration*, *look at it duration*, and *distance of mouse movement* are reliable indicators for more users than other indicators – 8 users out of 11. Over the data set of "all visits", the indicators were able to predict the most number of users' interests as well – 7 users out of 11. The *distance of mouse movement* was as accurate as indicators based on duration, and it can be the most practical indicator since it is simple to detect and is more robust than *active window duration* against the case of user's absence. If a user leaves a web page open and leaves the room, the *distance of mouse movement* will not be affected. For the *bookmark*, *save*, *print*, and *memo* indicators, more than 95% of the pages were correctly scored as "interested". When we divided the data set less than "interested" and more than or equal to "interested", "95% of the bookmarked web pages, 98% of the saved web pages, 100% of the printed web pages, and 98% of the memoed web pages belonged to the score of more than or equal to "interested". Our results also indicate that there was no indicator that was valid for all users. Depending on the user, an indicator may or may not be valid.

8.2. Ethical Issues in User Modeling

8.2.1. Privacy

An individual's right to privacy has always been an issue in user modeling. This is because the fact that the consequences for victims of privacy intrusions can be serious problems. Although the Internet is widely used nowadays, many users remain unfamiliar and skeptical about the level of security and privacy on the Internet. Great numbers of users are uncomfortable with "user profiling," a practice in which users' online movements are recorded. Much of this user anxiety is caused by the fact that users have no clear understanding regarding the rules that govern this practice, how extensive it is, what is recorded, and even how the information is used (Chiu, 2000). If the process of user profiling is explained, then user modeling will be more readily accepted.

This personalization method should be provided as an option of a web browser to a user along with full descriptions. Before building a user profile, the browser must accept the user's permission whether s/he wants to build the user profile from their bookmarks or the web pages collected by implicit interest indicator. This is a permission-based personalization tool. Furthermore, the profile can be stored in a client. If both the web log file of the user's behavior and the user profile remain in the user's computer, then s/he will feel their privacy is protected. The browser should provide complete controls over the profile to the user.

8.2.2. Confidence on the Results

This section briefly discusses the extent to which we can trust the profile generated by the DHC algorithm. For example, if the profile indicates a user is interested in "laptop computer", how much can we trust the result? The user profile is built out of a set of interesting web pages to a user. As previously explained, bookmarks or web pages detected by an implicit indicator can be used as the set. Since the set of interesting web pages can change over time, the user profile can change as well. The profiles should be rebuilt periodically. Then, are the interests of the user that appear over consecutive different periods more confident than the interests that occur only once? The results may also depend on how reliable the input data sets are. In order to answer the question, we may have to be able to measure the reliability of the set of interesting web pages. These questions are not easy for us to answer at this moment. It can be future work.

8.3. Limitation and Future Work

In our system there are several limitations.

- We did not analyze differences among the UIHs' obtained from various users because of the large numbers of web pages used in our experiments.
- The performance of the DHC algorithm varied depending on the articles selected. We believe this is because of the intrinsic characteristics in a document.
- The performance of VPF varied depending on the articles selected. We currently do not understand the reason for the variance in performance over different

articles. We assume it is due to the intrinsic characteristics of an article, because the human subjects' results are also different depending on the articles.

- Our experiment for desirable properties of a correlation function was limited to positive correlations for our web personalization since many applications depend on positive correlation. We will extend our analysis to negative correlation as well.
- The improvement of WS was not statistically significant because the precision values of Google had large variance.
- The reason for the low performance of some search terms might be because there is no relation between his/her bookmarks and the search terms. We may be able to relieve this problem by incorporating interesting web pages based on implicit interest indicators.
- Our approach of penalizing the index pages did not make much improvement in our initial experiments. We will examine this approach further in the future.
- Since WS showed higher performance for links after Top 5 than Google, we expect that our method may get higher performance with clustered search engines.
- A longer evaluation would give more accurate results for the *LookAtIt* indicator, since users would act more naturally after more than 1 or 2 hours of surfing.
- We can combine this indicator to an application for personalized web search results in the future. The collected interesting web pages for a user can be used for building a user interest hierarchy.

References

Note: Internet references current as of March 2005

Agrawal, R. and R. Srikant (1994), "Fast Algorithm for Mining Association Rules", *Proc. 20th Vary Large Data Base Conference*, 487-499.

Ahonen, H., Heinonen, O., Klemettinen, M. and A.I. Verkamo (1998), "Applying Data Mining Techniques for Descriptive Phrase Extraction in Digital Document Collections", *Proc. Advances in Digital Libraries Conference*, 2-11.

Albanese, M., Picariello, A., Sansone, C., and L. Sansone (2004), "Web Personalization Based on Static Information and Dynamic User Behavior", Proc. *6th annual ACM international workshop on Web information and data management*, 80-87.

Anderson, C.R. (2002), *A Machine Learning Approach to Web Personalization*, Ph.D. thesis. University of Washington, Department of Computer Science and Engineering. http://www.the4cs.com/~corin/research/pubs/thesis.pdf

Ardissono, L., Console, L., and I. Torre (1999), "Exploiting User Models for Personalizing News Presentations", *Proc. 2nd Workshop on Adaptive Systems and User Modeling on the WWW*.

Barzilay, R., McKeown, K., and M. Elhadad (1999), "Information Fusion in the Context of Multi-Document Summarization", *Proc. 37th Annual Meeting of the Association for Computational Linguistics*. http://www.cs.mu.oz.au/acl/P/P99/P99-1071.pdf

Bellegarda, J.R. (1998), "Exploiting Both Local and Global Constraints for Multi-Span Statistical Language Modeling", *Proc. Intl. Conf. On Acoustics, Speech, and Signal Processing*, IEEE press, 2, 677-680.

Bharat, K. and G.A. Mihaila (2001), "When Dxperts Agree: Using Non-Affiliated Experts to Rank Popular Topics", *Proc. 10th Intl. World Wide Web Conference*.

Billsus, D. and M.J. Pazzani (1999), "A Hybrid User Model for News Story Classification", *Proc. 7th International Conference on User Modeling*, Verlag, Wien-New York: Springer, 99-108.

Boyle, C. and A.O. Encarnacion (1994), "MetaDoc: An Adaptive Hypertext Reading System", *User Modeling and User-Adapted Interaction* 4, 1 (Jan.), 1-19.

Brin, S., Motwani, R., Page, L., and T. Winograd (1998), "What Can You Do with a Web in Your Pocket", In Bulletin of the IEEE Computer Society Technical Committee on Data Engineering.

Brusilovsky, P. and M.T. Maybury (2002), "From Adaptive Hypermedia to Adaptive Web", In P. Brusilovsky and M. T. Maybury (eds.), *Communications of the ACM* 45 (5), Special Issue on the Adaptive Web, 31-33.

Brusilovsky, P. and L. Pesin (1998), "Adaptive Navigation Support in Educational Hypermedia: An Evaluation of the ISIS Tutor", *Journal of Computing and Information Technology* 6, 1, 27-38.

Brusilovsky, P. (2001), "Adaptive hypermedia", *User Modeling and User Adapted Interaction*, Ten Year Anniversary Issue (Alfred Kobsa, ed.) 11 (1/2), 87-110. http://umuai.informatik.uni-essen.de/anniversary.html

Cadez, I., Heckerman, D., Meek, C., Smyth, P., and S. White (2000), "Visualization of Navigation Patterns of a Web Site Using Model-Based Clustering", *Proc. 6th International Conference on Knowledge Discovery and Data Mining*.

Chan, P.K. (1999), "A Non-Invasive Learning Approach to Building Web User Profiles", *In KDD-99 Workshop on Web Usage Analysis and User Profiling*, 7-12.

Cheeseman, P. and J. Stutz (1996), "*Bayesian Classification (AutoClass): Theory and Results*", Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, Menlo Park, Calif., 153-180.

Chen, L. and K. Sycara (1998), "Webmate: A Personal Agent for Browsing and Searching", *Proc. 2nd International Conference on Autonomous Agents*, pp.132-139.

Chiu, A.S. (2000), "The Ethics of Internet Privacy", http://web.tepper.cmu.edu/files/PDF_Document/c3f42085bfea4608914d62539d4f5579.pdf

Claypool, M., Le, P., Wased, M., and Brown, D., (2001) "Implicit Interest Indicators", *Proc. 6th international conference on Intelligent User Interfaces*, 33-40.

Cohen, W.W. (1995), "Fast Effective Rule Induction", *Proc. Twelfth International Conference*.

Cohen, W.W. (1998), "Joins that Generalize: Text Classification Using WHIRL", *Proc. 4th International Conference on Knowledge Discovery and Data Mining* (*KDD*-98).

Croft, W.B. and R. Das (1989), "Experiments with Query Acquisition and Use in Document Retrieval Systems", *Proc. 13th ACM SIGIR*.

Croft, W.B. and R.T. Thompson (1987), "I3R: A New Approach to the Design of Document Retrieval Systems", *Journal of the Americal Society for Information Science*, 38: 389-404.

Croft, W.B., Turtle, H.R., and D.D. Lewis (1991), "The Use of Phrases and Structure Queries in Information Retrieval", *ACM SIGIR Conference on Research and Development in Information Retrieval*, 32-45.

Croft, W.B. (2000), (editor) *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, Massachusetts, Kluwer Academic Publishers, 243.

Crotf, W.B. and R. Das (1989), "Experiments with Query Acquisition and Use in Document Retrieval Systems", *Proc. 13th ACM SIGIR*.

Cutting, D.R., Karger, D.R., Pedersen, J.O., and J.W. Tukey (1992), "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections", *Proc. 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Delaney, K.J. (2004), "Study Questions Whether Google Really is Better," *Wall Street Journal* (Eastern edition), New York, May 25, B.1. http://proquest.umi.com/pqdweb?RQT=309&VInst=PROD&VName=PQD&VType=PQD&sid=5&index=45&SrchMode=1&Fmt=3&did=000000641646571&clientId=15106

Deshpande, M. and G. Karypis (2001), "Selective Markov Models for Predicting Web-Page Accesses", *First SIAM International Conference on Data Mining*.

Eirinaki, M., Lampos, C., Paulakis, S., and M. Vazirgiannis, (2004a) "Web Personalization Integrating Content Semantics and Navigational Patterns", *Workshop on Web Information and Data Management*, 72 – 79.

Fagan, J.L. (1987), "Automatic Phrase Indexing for Document Retrieval", *Proc. 10th Annual ACM SIGIR Conference on Research & Development in Information Retrieval*, 91-101.

Fink, J., Kobsa, A., and A. Nill (1996), "User-oriented Adaptivity and Adaptability in the AVANTI project", *Proc. for the Web: Empirical Studies*, Microsoft Usability Group, Redmond (WA).

Fisher, D.H. (1987), "Knowledge Acquisition via Incremental Conceptual Clustering". *Machine Learning* 2, 139-172.

Frakes, W.B. and R. Baeza-Yates (1992), *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall.

Fu, X., Budzik, J., and K.J. Hammond (2000), "Mining Navigation History for Recommendation", *Proc. 2000 Conference on Intelligent User Interfaces*.

Gennari, J.H., Langley, P., and D. Fisher (1989), "Models of Incremental Concept Formation", *Artificial Intelligence*, 40, 11-61.

Goecks, J. and J. W. Shavlik (2000), "Learning Users' Interests by Unobtrusively Observing Their Normal Behavior", *Proc. ACM Intelligent User Interfaces Conference* (*IUI*), Jan 2000.

Goecks, J. and J.W. Shavlik (2000), "Learning Users' Interests by Unobtrusively Observing Their Normal Behavior", *Proc. ACM Intelligent User Interfaces Conference* (*IUI*), Jan.

Gokcay, D. and E. Gokcay (1995), "Generating Titles for Paragraphs Using Statistically Extracted Keywords and Phrases, Systems, Man and Cybernetics", *Proc. IEEE International Conference* on *Intelligent Systems for the 21st Century*, Vol. 4, 22-25, Oct.

Google co. (2004), Google. http://www.google.com/

Granka, L. A., Joachims, T., and G. Gay (2004), "Eye-tracking Analysis of User Behavior in WWW Search", *Proc. 27th annual international conference on Research and development in information retrieval*.

Gravano, L., Garcia-Molina, H., and A. Tomasic (1999), "Gloss: Text-source Discovery over the Internet", *ACM Transactions on Database Systems*, 24(2):229-264, June.

Grossman, D., Frieder, O., Holmes, D., and D. Roberts (1997), "Integrating Structured Data and Text: A Relational Approach", *Journal of the American Society for Information Science*, 48(2), February.

Guha, S., Rastogi, R., and K. Shim (1998), "CURE: An Efficient Clustering Algorithm for Large Databases", *Proc. ACM-SIGMOD Int. Conf. Management of Data* (*SIGMOD'98*), 73–84.

Guha S., Rastogi, R., and K. Shim (1999), "ROCK: A Robust Clustering Algorithm for Categorical Attributes", *Proc. 15th Int'l Conf. on Data Eng.*

Han, J. (2001), eds., *Data Mining: Concepts and Techniques*, San Francisco: Morgan Kaufmann Publishers, pp.338.

Harper, D.J. (1980), *Relevance Feedback in Document Retrieval Systems: An Evaluation of Probabilistic Strategies*, Ph.D. Thesis, Computer Laboratory, University of Cambridge.

Hartigan, J. (1975), *Clustering Algorithm*, John Wiley.

Haveliwala, T.H. (1999), "Efficient Computation of PageRank", Technical Report, Stanford University Database Group. http://dbpubs.stanford.edu/pub/1999-31

Haveliwala, T.H. (2002), "Topic-sensitive PageRank", *Proc. 11th Intl. World Wide Web Conference*, Honolulu, Hawaii, May.

Herlocker, J., Konstan, J., Borchers, A., and J. Riedl (1999), "An Algorithmic Framework for Performing Collaborative Filtering", *Proc. 1999 Conference on Research and Development in Information Retrieval*.

Hilderman, R. and H. Hamilton (2001), "Evaluation of Interestingness Measures for Ranking Discovered Knowledge", *Proc. 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Hoerding, T. (1999), "A Temporary User Modeling Approach for Adaptive Shopping on the Web", *Proc. 2nd Workshop on Adaptive Systems and User Modeling on the WWW*.

Huang, S., An, A., and N. Cercone (2002), "Comparison Of Interestingness Functions for Learning Web Usage Patterns", *Proc. eleventh international conference on Information and knowledge management*, 617-620.

Hull, D.A. (1994), *Information Retrieval Using Statistical Classification*, PhD thesis, Stanford University, Statistics.

Intel Inc. (2001), "Open Source Computer Vision Library", *Reference Manual*, Intel Corporation, 2-1~2-2.

Jaroszewicz, S. and D. A. Simovici (2004), "Interestingness of Frequent Item sets Using Bayesian Networks as Background Knowledge", *Proc. 2004 ACM SIGKDD international conference on Knowledge discovery and data mining.* 2004, 178 – 186.

Jeh, G. and J. Widom (2003), "Scaling Personalized Web Search", *Proc. 12th Intl. Conference on World Wide Web*, Budapest, Hungary, 20-24, May.

Joachims, T., Freitag, D., and T. Mitchell (1997), "Web Watcher: A Tour Guide for the World Wide Web", *Proc. 15th International Joint Conference on Artificial Intelligence*, 770-775.

Johansson, C. (1996), "Good Bigrams", *Proc. COLING-96*, 592-597.

Jung, K. (2001) *Modeling Web User Interest with Implicit Indicators*, Master Thesis, Florida Institute of Technology.

Kamber, M. and R. Shinghal (1996), "Evaluating the Interestingness of Characteristic Rules", *Proc. 2nd International Conference on Knowledge Discovery and Data Mining*, 263-266, Portland, Oregon.

Kaplan, C., Fenwick, J., and J. Chen (1993), "Adaptive Hypertext Navigation Based on User Goals and Context", *User Modeling and User-Adapted Interaction* 3, 3, 193-220.

Karypis, G., Han, E., and V. Kumar (1999), "CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling", *IEEE Computer*.

Kaufman, L. and P.J. Rousseeuw (1990), *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York.

Kim, D., Atluri, V., Bieber, M., Adam, N., and Y. Yesha (2004), "A clickstream-based collaborative filtering personalization model: towards a better performance", *Proc. 6th annual ACM international workshop on web information and data management*, 88-95.

Kim, H. and P.K. Chan (2003), "Learning Implicit User Interest Hierarchy for Context in Personalization", *Proc. International Conference on Intelligent User Interfaces*, 101-108.

Kim, H. and P.K. Chan (2004), "Identifying Variable-Length Meaningful Phrases with Correlation Functions", *International Conference on Tools with Artificial Intelligence* (*ICTAI*), IEEE press, 30-38.

Kim, H. and P. K. Chan (2005), "Implicit Indicator for Interesting Web Pages", Technical Report CS-2005-05, Florida Institute of Technology.

Kim, J., Oard, D.W., and K. Romanik, (2001), "Using Implicit Feedback for User Modeling in Internet and Intranet Searching", Technical Report, College of Library and Information Services, University of Maryland, May.

Li, W.S., Vu, Q., Agrawal, D., Hara, Y., and H. Takano (1999), "PowerBookmarks: A System for Personalizable Web Information Organization, Sharing, and Management", *Proc. 8th International World Wide Web Conference*, Toronto, Canada.

Liberman, H. (1995), "Letizia: An Agent that Assists Web Browsing", *Proc. 14th International Joint Conference on Artificial Intelligence*.

Lima, E.F. and J.O. Pedersen (1999), "Phrase Recognition and Expansion for Short, Precision-Biased Queries Based on a Query Log", *Proc. SIGIR*.

Lind, D.A., Marchal, W.G., and R.D. Mason (2002), *Statistical Techniques in Business & Economics 11th edition*, McGraw-Hill Irwin, 377-412.

Liu, F., Yu, C., and W. Meng (2002), "Personalized Web Search by Mapping User Queries to Categories", *Proc. CIKM'02*, ACM Press, Virginia, USA.

Maarek, Y.S. and I.Z. Ben-Shaul (1996), "Automatically Organizing Bookmarks Per Contents", *Proc. 5th International World Wide Web Conference*.

Mahoney, M.V. and P.K. Chan (2003), "Learning Rules for Anomaly Detection of Hostile Network Traffic", *Proc. 3rd International Conference on Data Mining* (*ICDM*).

Mannila, H. and H. Toivonen (1996), "Discovering Generalized Episodes Using Minimal Occurrences", *Proc. Knowledge Discovery and Data Mining*.

Milligan, G.W. and M.C. Cooper (1985), "An Examination of Procedures for Detecting the Number of Clusters in a Data Set", *Psychometrika*, 50, 159-59.

Mitchell, T. (1997), *Machine Learning*, McGraw-Hill, 81-126 and 154-199.

Mitra, M., Buckley, C., Singhal, A., and C. Cardie (1997), "An Analysis of Statistical and Syntactic Phrases", *Proc. RIAO-97, 5th International Conference*.

Mladeni, D. (2000), "Machine Leaning for Better Web Browsing", *AAAI Spring Symposium on Adaptive User Interfaces*.

Mobasher, B., Cooley, R., and J. Srivastave (1999), "Creating Adaptive Web Sites through Usage-Based Clustering of URLs", *Proc. 1999 IEEE Knowledge and Data Engineering Exchange Workshop*, 19-25.

Mobasher, B., Dai, H., Luo, T., Sun, Y., and J. Zhu (2000), "Combining Web Usage and Content Mining for More Effective Personalization", *Proc. International Conference on E-Commerce and Web Technologies* (*ECWeb*).

Ng, R., and J. Han (1994), "Efficient and Effective Clustering Method for Spatial Data Mining", *Proc. Int. Conf. on Very Large Data Bases* (*VLDB'94*), 144–155.

Oard, D. and J. Kim. (1998) "Implicit Feedback for Recommendation Systems", *Proc. the AAAI Workshop on Recommendation Systems*, July.

Page, L., Brin, S., Motwani, R., and T. Winograd (1998), "The PageRank Citation Ranking: Bringing Order to the Web", Technical Report, Stanford University Database Group, 1998. http://citeseer.nj.nec.com/368196.html

Pazzani, M. and D. Billsus (1997), "Learning and Revising User Profiles: The Identification of Interesting Web Sites", *Machine Learning*, 27(3), 313-331.

Pazzani, M. and D. Billsus (1999a), "Adaptive Web Site Agents". *Proc. 3rd International Conference, Autonomous Agents*.

Pazzani, M. and D. Billsus (1999b), "Evaluating Adaptive Web Site Agents", *Proc. Workshop on Recommender Systems Algorithms and Evaluation, 22nd International Conference on Research and Development in Information Retrieval*.

Pazzani, M., Muramatsu, J., and D. Billsus (1996), "Syskill & Webert: Identifying Interesting Web Sites". *Proc. National Conference on Artificial Intelligence*, 54-61.

Pazzani, M. and D. Billsus (1997), "Learning and Revising User Profiles: The Identification of Interesting Web Sites", *Machine Learning*, 27(3), 313-331.

Perkowitz, M. and O. Etzioni (1997), "Adaptive Web Sites: an AI Challenge", *Proc. 15th International Joint Conference on Artificial Intelligence*.

Perkowitz, M. and O. Etzioni (1998), "Adaptive Web Sites: Automatically Synthesizing Web Pages", *Proc. AAAI98*.

Perkowitz, M. and O. Etzioni (2000), "Towards Adaptive Web Sites: Conceptual Framework and Case Study", *Artificial Intelligence* 118, 245-275.

Perkowitz, M. (2001), *Adaptive Web Sites: Cluster Mining and Conceptual Clustering for Index Page Synthesis*, PhD thesis, University of Washington, Computer Science and Engineering.

Piatetsky-Shapiro, G. (1991), "Discovery, Analysis and Presentation of Strong Rules", In G. Piatetsky-Shapiro and W. Frawley, editors, *Proc. Knowledge Discovery in Database*, 2299-248. MIT Press, Cambridge, MA.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and J. Riedl (1994) "GroupLens: An open architecture for collaborative filtering of netnews". In Richard K. Faruta and Christine M. Neuwirth, editors, *Proc. Conference on Computer Supported Cooperative Work*, 175-186, ACM, October

Rasmussen, E. (1992), Clustering algorithms. *Information Retrieval: Data Structures and Algorithms*, W.B. Frakes and R. Baeza-Yates, Editors, Prentice Hall, Englewood Cliffs, NJ.

Robertson, S.E. (1981), "The Methodology of Information Retrieval Experiment", In: Sparck Jones, editor, *Information Retrieval Experiment*, London: Butterworths, 9-31.

Rosenfeld, R. (1994), *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, PhD thesis, Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Russell, S. and P. Norvig (1995), eds., *Artificial Intelligence: A Modern Approach*, Prentice Hall, pp.74.

Salton, G. and R.G. Waldstein (1978), "Term Relevance Weights in On-Line Information Retrieval", *Information Processing and Management*, 14, 29-35.

Schraefel, M.C., Modjeska, D., Wigdor, D., and Y. Zhu (2002), "Hunter Gatherer: Interaction Support for the Creation and Management of Within-Web-page Collections", *Proc. 11th International World Wide Web Conference*.

Shahabi, C. and F. Banaei-Kashani (2003), "Efficient and Anonymous Web-Usage Mining for Web Personalization", *INFORMS Journal on Computing-Special Issue on Data Mining*, 15 (2), Spring.

Shardanand, U. and P. Maes (1995), "Social Information Filtering: Algorithms for Automating Word Of Mouth", *Proc. ACM CHI Conference*.

Stefani, A. and C. Strapparava (1999), "Exploiting NLP Techniques to Build User Model for Web Sites: the Use of WorldNet in SiteIF Project", *Proc. 2nd Workshop on Adaptive Systems and User Modeling on the WWW*.

Sutton, R.S. and A.G. Barto (1998), *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

Tan, P. and V. Kumar (2000), "Interestingness Measure for Association Patterns: A Perspective*", *Proc. KDD*.

Tan, P., Kumar, V. and J. Srivastava (2002), "Selecting the Right Interestingness Measure for Association Patterns", *Porc. ACM SIGKDD*.

Turpin, A. and A. Moffat (1999), "Statistical Phrases for Vector-space Information Retrieval", *Proc. SIGIR*, 309-310.

Ungar, L.H. and D.P. Foster (1998), "Clustering Methods for Collaborative Filtering", *AAAI Workshop on Recommendation Systems*. http://citeseer.nj.nec.com/cache/papers/cs/18784/http:zSzzSzwww.cis.upenn.eduzSzdataminingzSzPublicationszSzclust.pdf/ungar98clustering.pdf

van Rijsbergen, C.J. (1979), *Information Retrieval*, Butterworths, London, 68–176.

Vivisimo co. (2005), Vivisimo. http://www.vivisimo.com

Voorhees, E.M. (1986), "Implementing Agglomerative Hierarchical Clustering Algorithms for Use in Document Retrieval", *Information Processing & Management*, 22 (6), 465-476.

Watson, A. and M. A. Sasse, (1998) "Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications", *Proc. ACM Multimedia Conference*, 55-60.

Webb, G.I., Pazzani, M.J., and D. Billsus (2001), "Machine Learning for User Modeling", *User Modeling and User-Adapted Interaction* 11: 19-29.

Webb, G.I. (1993), "Feature Based Modelling: A Methodology for Producing Coherent, Consistent, Dynamically Changing Models of Agents Competency", *Proc. 1993 World Conference on Artificial Intelligence in Education* (*AACE*), 497-504. http://www.cm.deakin.edu.au/~webb/

Weber, G. and M. Specht (1997), "User Modeling and Adaptive Navigation Support in WWW-bases Tutoring Systems", *Proc. 6th International Conference on User Modeling* (*UM97*).

Wexelblat, A. and P. Maes (1997), "Footprints: History-Rich Web Browsing", *Proc. Conference on Computer-Assisted Information Retrieval* (*RIAO*), 75-84.

Widmer, G. and M. Kubat (1996), "Learning in the Presence of Concept Drift and Hidden Contexts", *Machine Learning*. http://citeseer.nj.nec.com/cache/papers/cs/15513/http:zSzzSzwww.cacs.usl.eduzSz~mkubatzSzpublicationszSzgermljfinal.pdf/widmer96learning.pdf

Willet, P. (1988), "Recent Trends in Hierarchical Document Clustering: A Critical Review", *Information Processing and Management*, 24, 577-597.

Wu, H. and D. Gunopulos (2002), "Evaluating the Utility of Statistical Phrases and Latent Semantic Indexing for Text Classification", *IEEE International Conference on Data Mining*, 713-716.

Yahoo (2003), Yahoo mail. http://edit.yahoo.com/config/eval_register?.v=&.intl=&new=1&.done=&.src=ym&.partner=&.p=&promo=&.last=

Yan, T.W., Jacobsen, M., Garcia-Molina, H., and U. Dayal (1996), "From User Access Patterns to Dynamic Hypertext Linking", *Proc. 5th International World Wide Web Conference*.

Zamir, O. and O. Etzioni (1998), "Web Document Clustering: A Feasibility Demonstration", *SIGIR Conference on Research and Development in Information Retrieval*, 46-54.

Zamir, O. and O. Etzioni (1999), "Groper: A Dynamic Clustering Interface to Web Search Results", *Computer Networks* 31, 1361-1374.

Zhang, T., Ramakrishnan, R. and M. Livny (1996), "BIRCH: An Efficient Data Clustering Method for Very Large Databases", *Proc. ACM SIGMOD Intl. Conf. on Management of Data*.

Zukerman, I., Albrecht, D.W., and A.E. Nicholson (1999), "Predicting Users' Requests on the WWW", *Proc. 7th Intel. Conference on User Modeling* (*UM*), Banff, Canada. 275-284

Appendix

Appendix 1. Correlation functions

	Name	Formula
1	ϕ-coefficient (Tan et al., 2002) (Coef)	(AB - (A × B)) / Sqr(A × B × (1 - A) × (1 - B))
2	Goodman-Kruskal's (Tan et al., 2002) (Good)	(MAX(AB, AB′) + MAX(A′B, A′B′) + MAX(AB, A′B) + MAX(AB′, A′B′) - MAX(A, A′) - MAX(B, B′)) / (2 - MAX(A, A′) - MAX(B, B′))
3	Odds ratio (Tan et al., 2002) (Odds)	D((AB × A′B′), (AB′ × A′B))
4	Yule's Q (Tan et al., 2002) (YulQ)	(AB × A′B′ - AB′ × A′B) / (AB × A′B′ + AB′ × A′B)
5	Yule's Y (Tan et al., 2002) (YulY)	(Sqr(AB × A′B′) - Sqr(AB′ × A′B)) / (Sqr(AB × A′B′) + Sqr(AB′ × A′B))
6	Kappa (k) (Tan et al., 2002) (Kapp)	(AB + A′B′ - (A × B) - (A′ × B′)) / (1 - (A × B) - (A′ × B′))
7	Mutual Information (Tan et al., 2002) (Mutu)	(AB × log2(AB / (A × B)) + AB′ × log2(AB′ / (A × B′)) + A′B × log2(A′B / (A′ × B)) + A′B′ × log2(A′B′ / (A′ × B′))) / (MIN(-(A × log2(A) + A′ × log2(A′)), -(B × log2(B) + B′ × log2(B′))))
8	J-Measure (Tan et al., 2002) (JMea)	MAX(AB × log2(P(B|A) / B) + AB′ × log2(P(B′|A) / B′), AB × log2(P(A|B) / A) + A′B × log2(P(A′|B) / A′))
9	Gini index (Tan et al., 2002) (Gini)	MAX(A(pow(P(B|A),2) + pow(P(B′|A),2)) + A′(pow(P(B|A′),2) + pow(P(B′|A′),2)) - pow(B,2) - pow(B′,2), B(pow(P(A|B), 2) + pow(P(A′|B), 2)) + B′(pow(P(A|B′),2) + pow(P(A′|B′),2)) - pow(A,2) - pow(A′,2))
10	Support (Tan et al., 2002) (Supp)	AB
11	Confidence (Tan et al., 2002) (ConMa)	MAX(P(B|A), P(A|B))
12	Laplace (Tan et al., 2002) (Lapl)	MAX((100 × AB + 1) / (100 × A + 2), (100 × AB + 1) / (100 × B + 2))
13	Conviction (Tan et al., 2002) (Conv)	MAX((A × B′) / AB′, (B × A′) / A′B)
14	Interest (Tan et al., 2002) (Inte)	AB / (A × B)
15	Cosine (Tan et al., 2002) (Cosi)	AB / Sqr(A × B)
16	Piatetsky-Shapiro's (Tan et al., 2002) (Piat)	AB - A × B
17	Certainty Factor (Tan et al., 2002) (Certa)	MAX((P(B|A) - B) / (1 - B), (P(A|B) - A) / (1 - A))
18	Added Value (Tan et al., 2002) (Added)	MAX(P(B|A) - B, P(A|B) – A)
19	Collective strength (Tan et al., 2002) (Coll)	((AB + A′B′) / (A × B + A′ × B′)) × ((1 - A × B - A′ × B′) / (1 - AB - A′B′))
20	Jaccard (Tan et al., 2002) (Jacc)	AB / (A + B - AB)
21	Klosgen (Tan et al., 2002) (Klos)	Sqr(AB) × MAX(P(B|A) - B, P(A|B) - A)
22	MI (Tan et al., 2002)	Log2(AB / (A × B))
23	STC_MIN (Zamir and Etzioni, 1998) (StcMi)	MIN(P(B|A), P(A|B))
24	EMI (Chan, 1999)	AB × log(AB / (A × B)) + AB′ × log(AB′ / (A × B′)) + A′B × log(A′B / (A′ × B)) + A′B′ × log(A′B′ / (A′ × B′))
25	AEMI4 (Chan, 1999)	AB × log(AB/ A × B) - AB′ × log(AB′/ A × B′) - A′B × log(A′B/ A′ × B) + A′B′ × log(A′B′/ A′ × B′)
26	DMAX	AB × MAX(P(B|A), P(A|B))

27	DMI	$AB \times \log2(AB / (A \times B))$
28	AEMI3	$AB \times \log(AB/ A \times B) - AB' \times \log(AB'/ A \times B') - A'B \times \log(A'B/ A' \times B)$
29	dMIN	$AB \times MIN(P(B\|A), P(A\|B))$
30	dMIN2	$1 + AB \times \log(MIN(P(B\|A), P(A\|B)))$
31	NegativeCosine (Ahonen et al., 1998) (NegCos)	$(1 - AB) / Sqr((1 - A) \times (1 - B))$
32	MutualConfidence (Ahonen et al., 1998) (MuConf)	$(AB / A + AB / B) / 2$