« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 1»

Appl Intell
DOI 10.1007/s10489-007-0056-0

Learning implicit user interest hierarchy for context in personalization

Hyoung-Rae Kim · Philip K. Chan

Received: 10 April 2007 / Accepted: 12 April 2007

Abstract 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 terms (topics) into a hierarchy where more general interests are represented by a larger set of terms. 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. Our experiment indicates that DHC with the Augmented Expected Mutual Information (AEMI) correlation function and MaxChildren threshold-finding method built more meaningful UIHs than the other combinations on average; using words and phrases as features improved the quality of UIHs.

Keywords Clustering algorithm · Correlation function · User interest hierarchy · User modeling · User profile

H.-R. Kim (✉)
Korea Employment Information Service, Information Strategy Team, 77-11 Mullae-dong 3-ga, YeongDeungPo-gu, Seoul, South Korea
e-mail: goddoes8@gmail.com

P.K. Chan
Department of Computer Sciences, Florida Institute of Technology, 150 West University Blvd., Melbourne, FL 32901, USA
e-mail: pkc@cs.fit.edu

1 Introduction

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.

More general interests can correspond to passive interests, while more specific interests correspond to active interests. During a browsing session, general interests are in the back of one's mind, while specific interests are the current foci. Unlike News Dude [3], which generates a long-term and a short-term model, we model a continuum of general to specific interests. We believe identifying the appropriate context underlying a user's behavior is important in more accurately pinpointing her/his interests.

The web is not static—new documents and new words/phrases are created every day. Most clustering methods cluster objects (documents) [6, 7, 25, 27]. This representation is inadequate in a dynamic environment like the web. Suffix Tree Clustering [28] does not rely on a fixed vector of word features in clustering documents. We use a similar approach—instead of clustering documents, we cluster features (terms) in the documents; documents are then assigned to the clusters. Terms are defined as words and phrases. Consider how a librarian forms a taxonomy of subjects for all the books in the library. She would first identify the subject(s)

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

of a book (e.g., Operating Systems (OS), Programming Languages (PL), Statistics (Stats), Calculus (Cal)) and then create a taxonomy of the subjects (e.g., group OS and PL under CS, and Stats and Cal under Math). Finally, books are categorized according to the taxonomy, where not all terms in the books are in the book catalog system. As the book catalog system is hierarchical, we propose to model general and specific interests (web browsing interests of a user) with a concept hierarchy called User Interest Hierarchy (UIH), while suffix tree clustering (STC) provides flat clusters.

Most search engines are not sensitive to a user's interests. An improved interface for the user would rank results according to the user's profile [13]. A UIH represents the user's specific as well as general interests, which can help rank results returned by a search engine. Pages that match the more specific interests receive a higher score than those that only match the more general interests. Furthermore, the UIH provides a context to disambiguate words that could have multiple meanings in different contexts. For example, "java" is likely to mean the programming language, not the coffee, for a UIH that is learned from a user who has been reading computer science related pages. This helps a user in searching relevant pages on the web.

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: it takes time and effort to specify 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 represent more general interests. For example, a graduate student in computer science is looking for a research paper in web personalization. The short-term specific interest is web personalization, but the general interest is computer science. The web pages the student is interested in 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 and duration of interest.

The main objective of this research is to build UIH's that capture general to specific interests without the user's involvement (implicitly). We propose a divisive hierarchical clustering (DHC) algorithm that constructs such a hierarchy. We believe our approach has significant benefits and possesses interesting challenges. We can improve the UIH by using phrases in addition to words. A term composed of 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 used phrases collected by a variable-length phrase-finding algorithm (VPF) [12].

The main contributions of this work 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 web pages bookmarked by a user;
- we devise a divisive graph-based hierarchical clustering algorithm (DHC), which constructs a UIH by grouping terms (topics) into a hierarchy instead of the flat cluster used by STC;
- DHC automatically finds the threshold for clusters of terms (words and phrases) whereas STC needs to specify the threshold;
- we use a more sophisticated correlation function, AEMI, than STC's conditional probability;
- our experimental results indicate that 64% of the generated UIH's are quite meaningful.

We also observed that DHC with an AEMI (Augmented Expected Mutual Information) correlation function and MaxChildren threshold-finding method made a more meaningful UIH than the other combinations.

Section 2 of this paper discusses related work in building the UIH; Sect. 3 introduces user interest hierarchies (UIH's); Sect. 4 details our approach towards building implicit UIH's; Sect. 5 discusses our empirical evaluation regarding the meaningfulness of UIH; Sect. 6 presents and analyzes generated UIHs; Sect. 7 summarizes our findings and suggests possible future work.

2 Related work

We discuss related work in two areas: user profiles and clustering algorithms. Since we are proposing a new representation of a user profile, we will review previous representations of user profiles. The DHC algorithm for building a UIH is a divisive hierarchical clustering algorithm. We will explain why we need to devise a new clustering algorithm by reviewing relevant clustering algorithms. Note that building UIH is different from other methods in document clustering since they take a large corpus of labeled (e.g., news categories) documents as input and then cluster the documents [19].

User profiles A user profile can be built based on the user's behavior, the contents of a web page, or both. A human behavior based user model can be learned by observing the user's actions such as web log file, path, click, downloads, or frequency. Pazzani and Billsus [18] 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

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

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. Mobasher et al. [17] 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 and Webert [19] is an intelligent agent that learns user profiles. After identifying informative words from web pages to use as Boolean features, it learns a Naive 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. [17] 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. Trajkova and Gauch [24] build user profiles automatically from the web pages visited by a user without user intervention. Their work focuses on improving the accuracy of the user profile based on concepts from a predefined ontology. The experimental results show that the user profile can achieve average accuracy of 69% when no concepts are pruned.

Our method in this paper is only concerned with the text but allows overlapping clusters, since once we get a user profile based on contents, we can extend it to combining human behavior based methods. A news agent called News Dude [3], 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 Naive Bayesian classifier for long-term interests.

Clustering algorithms STC [28] is a document-clustering algorithm using a suffix tree. By using a suffix tree with words that are not too few (3 or less) or too many (more than 40% of the collection), STC (suffix tree clustering) finds phrases and document frequency of terms (words/phrases). After finding the terms, STC calculates the similarity between terms using the document frequency and MIN function. The connection between terms is determined by the strength of the similarity values. STC applies graph-based partitioning to group the terms connected only once, thus results in flat clusters. Given the desirable number of clusters, AutoClass [5] estimates the interclass probability (an object belonging to a certain cluster) and intraclass probability (the object's attribute values if the object belongs to the cluster) to calculate the probabilities of an object being a member of the different clusters. That is, each object does not belong to exactly one cluster, which is the case for most clustering algorithms. Furthermore, AutoClass does not generate hierarchical clusters. The disadvantages of flat clusters are they cannot represent different abstraction levels of clusters.

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 [26]. 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. These algorithms are very sensitive to the stopping criterion. Several stopping criteria for AHC algorithms have been suggested, but they are typically predetermined constants—one common stopping criterion is the desired number of clusters [8, 15]. 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 [28]. 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. COBWEB [8] is an incremental system for hierarchical conceptual clustering, which carries out a hill-climbing search through a space of hierarchical classification schemes. The heuristic evaluation measure used to guide the search is the similarity of objects within the same class and dissimilarity of objects in different classes. This measure uses the expected number of correct guesses for attributes' values. Each cluster records the probability of each attribute and value, and the probabilities are updated every time an object is added to a cluster. Since our problem domain has only one attribute (document frequency of terms), COBWEB would group objects with the same attribute value in one cluster and no further divisions are necessary. Thus, COBWEB generates flat clusters.

For measuring the similarity between a pair of terms, we apply AEMI instead of MIN unlike STC (more details in Sect. 4.2). Suffix tree clustering (STC) sets a threshold to differentiate strong from weak connections between a pair of terms; weak connections are removed by applying MaxChildren threshold finding method in our method (in Sect. 4.3). We recursively group the sub-clusters and build hierarchical clusters instead of flat clusters. Our DHC algorithm can generate multiple branches from one node depending on the data (instead of only two branches), which is the advantage of using a graph-partitioning technique. Another difference of our algorithm from other AHC/DHC algorithms is that all objects in the root node may not be in the child nodes. It is like the book catalog system, where all terms in the books are not in the catalog.

3 Problem

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 terms while towards the leaves, more specific (active) interests are represented by smaller clusters of terms. To generate a UIH for a user, our clustering algorithm (details in Sect. 4) accepts a set of web pages bookmarked by the user as input. That is, the input of DHC is documents that are interesting to a user (e.g., bookmarks). We, however, are not clustering documents but terms in documents. We use the words and phrases in a web page and ignore link or image information. The web pages are stemmed and filtered by ignoring the most common words listed in a stop list (called "function words") which are usually non-content words such as conjunctions, determiners, and prepositions [21, 23]. The phrases are extracted by variable-length phrase-finding algorithm [12]. These processes are depicted in Fig. 1.

Table 1 contains a sample data set. Numbers on the left represent individual web pages; the content has words

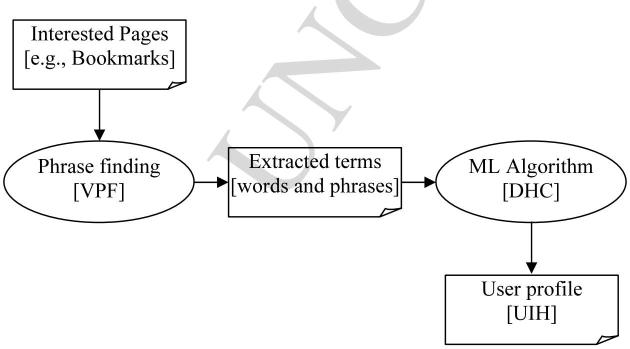


Fig. 1 Process diagram

stemmed and filtered through the stop list. These words in the web pages can be represented by a UIH as shown in Fig. 2. Each node (cluster) contains a set of words. The root node contains all words that exist in a set of web pages. The specificity of the root node may depend on the number of web pages. As the set of interesting web pages to a user increases, the root node becomes more general. 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) cannot. Words in this node (in the dashed box) 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.

Since one can easily identify phrases such as "machine learning" and "searching algorithm" 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 a feature in the UIH, each cluster will be enriched because phrases are more specific than words. For example, a user is interested in "java coffee" and "java language". The word "java" will be in the parent cluster of both "coffee" and

Table 1 Sample data set

Web age	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 checking
10	ai searching algorithm constraint reasoning forward checking

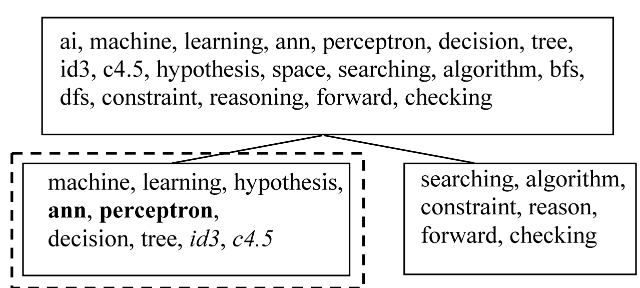


Fig. 2 Sample user interest hierarchy

"language". Each child cluster would contain only "coffee" or "language", which is relatively less useful when not in combination with "java".

Note that our approach can *indirectly* cluster pages where pages may belong to multiple clusters—overlapping clusters of pages. Instead of directly clustering the original objects (web pages), this indirect cluster method first cluster features (words) of the objects and then the objects are assigned to clusters based on the features in each cluster. Since a document can have terms in different clusters, 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 paper is on devising and evaluating algorithms for this step. We call our hierarchical clustering of features a UIH, because it represents a user's general to specific interests.

4 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 terms into smaller clusters, which represent more related terms. We assume terms occurring close to each other (within a window size) are related to each other. We investigate correlation functions that measure how closely two terms are related in Sect. 4.2. We also study techniques that dynamically locate a threshold that decides whether two terms are strongly related or not in Sect. 4.3. If two terms are determined to be strongly related to each other, they will be in the same cluster; otherwise, they will be in different clusters.

4.1 Algorithm

Our algorithm is a divisive graph-based hierarchical clustering method (DHC), that recursively divides clusters into child clusters until it meets the stopping conditions. We set a minimum number of terms (MinClusterSize) in a cluster as the stopping condition. In preparation for our clustering algorithm, we extract terms from web pages that are interesting to the user by filtering them through a stop list, stemming them [21, 23], and adapting variable-length phrase-finding (VPF) algorithm [12]. Figure 3 illustrates the pseudo code for the DHC algorithm. Using a correlation function, we calculate the strength of the relationship between a pair of terms in line 1. The WindowSize is the maximum distance (in number of words) between two related terms in calculating their correlation value. After calculating a threshold to differentiate strong correlation values from weak correlation in line 2, we remove all weak correlation values in line 5. The FINDTHRESHOLD is a method that calculates the cutoff value for determining strong and weak correlation values. We then build a weighted undirected graph with each vertex representing a term and each weight denoting the correlation between two terms. Since related terms are more likely to appear in the same document than unrelated terms, we measure co-occurrence of terms in a document. Given the graph, called a CorrelationMatrix, the clustering algorithm recursively partitions the graph into subgraphs, 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

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

Fig. 3 DHC algorithm

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

more computation is required). 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 terms in the cluster falls below a predetermined threshold.

Suppose we built a weighted undirected graph with the running example in Table 1 where each vertex represents a term and each weight (value) denotes the correlation value. The undirected graph can be depicted as shown in Fig. 4(a)—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)—those edges whose value is low are hidden to reduce the complexity of the graph. 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 Fig. 4(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 Fig. 4(c). Since DHC recursively partitions the graph into subgraphs, called Clusters, the final result becomes hierarchical clusters as shown in Fig. 4(d). Note that the edge between "ann" and "learning" does not appear in (a) and (b). It appears only in (c) and (d) after the recalculation of the correlation values. This happens because when we calculated the edge with the whole terms, the edge was weak. When we calculated the correlation value for each sub cluster, however, the correlation value became high in its sub cluster.

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 terms (the number of words between two terms) 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 (Sect. 4.2) and threshold-finding method (Sect. 4.3) greatly influence the clustering algorithm, and are discussed next.

4.2 Correlation functions

The correlation function calculates how strongly two terms (words or phrases) are related. Since related terms are likely to be closer to each other than unrelated terms, we assume two terms 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 terms co-occur if they are in the same document. These functions are used in `CalculateCorrelationMatrix` function in Fig. 3.

4.2.1 AEMI

We use AEMI (Augmented Expected Mutual Information) [4] 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 [4] demonstrates that AEMI could find more meaningful multi-word phrases than MI or EMI. Concretely, consider variables A and B in $\mathrm{AEMI}(A,B)$ are the events for the two terms (a and b), where the capital A and B are variables and lowercase a and b are the instances. $P(A=a)$ is the probability of a document containing a term of a and $P(A=\bar{a})$ is the probability of a document not having term a. For example, if out of 100 documents 5 documents contain the term of a, then $P(A=a)$ is 0.05 and $P(A=\bar{a})$ is 0.95. $P(B=b)$ 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. $\mathrm{AEMI}(A,B)$ is defined as:

$$\begin{aligned}\mathrm{AEMI}(A,B) &\\ &= P(a,b)\log\frac{P(a,b)}{P(a)P(b)}\\ &\quad - \sum_{(A=a,B=\bar{b})(A=\bar{a},B=b)} P(A,B)\log\frac{P(A,B)}{P(A)P(B)}. \qquad (1)\end{aligned}$$

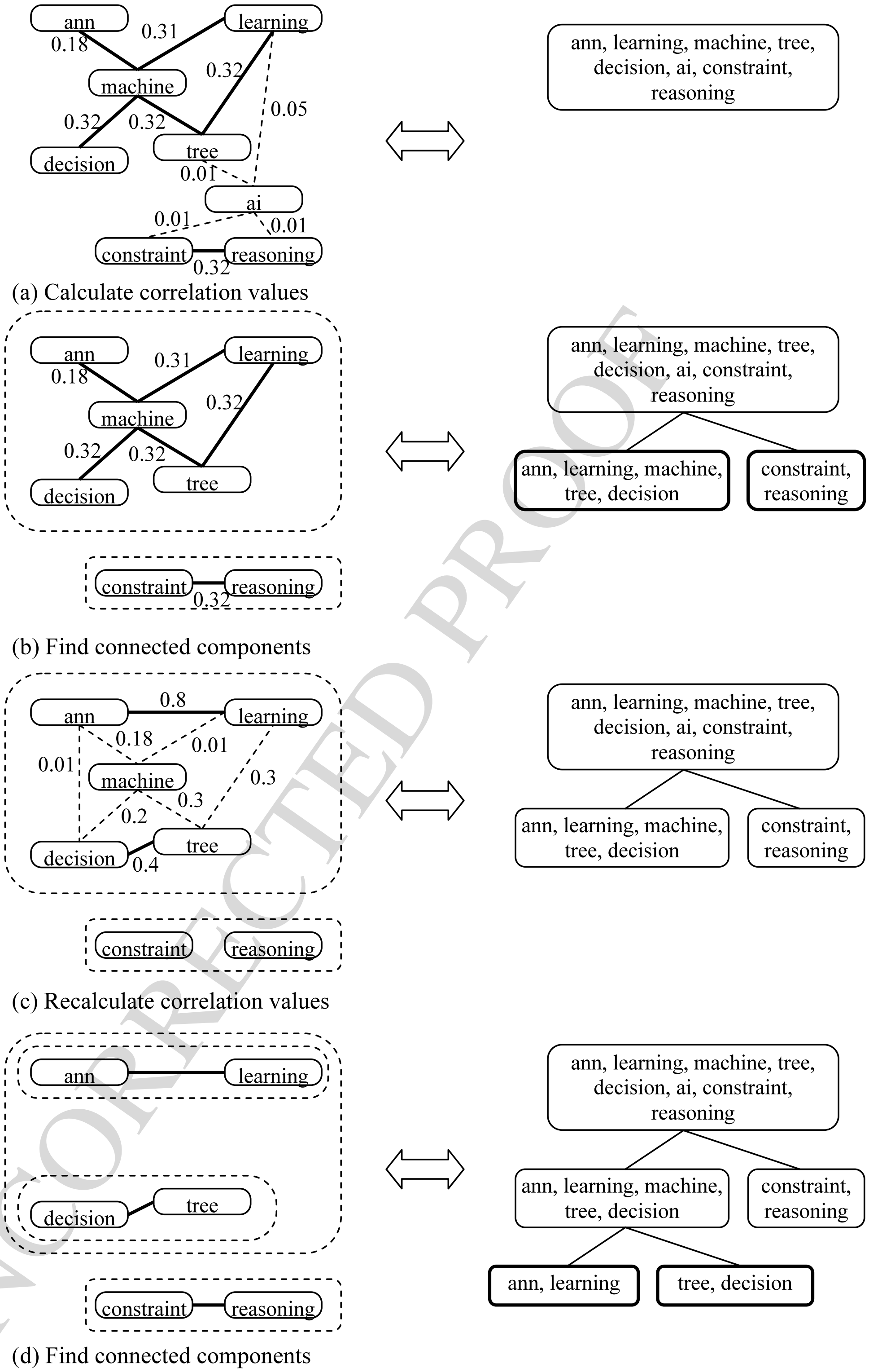
The first term computes supporting evidence that a and b are related and the second term calculates counter-evidence. Using our running example in Fig. 2, Table 2 shows a few examples of how AEMI values are computed. The AEMI value between 'searching' and 'algorithm' is 0.36, which is higher than the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

Table 2 AEMI values

$P(a)$	$P(\bar{a})$	$P(b)$	$P(\bar{b})$	$P(ab)$	$P(\bar{a}b)$	$P(a\bar{b})$	$\mathrm{AEMI}(a,b)$
$a = searching, b = algorithm$							
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

« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 7»

Fig. 4 An example of DHC algorithm



4.2.2 *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 term calculates the number of documents that contain the term. Terms 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)

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

Table 3 AEMI-SP values

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

measures how informative a term is in characterizing the content. While involving the IDF, we adapt sigmoid function in order to emphasize more specific (informative) terms. The adjusted sigmoid function is called SP (specificity).

We estimate the probability of document frequency of a term so that we can scale the quantity between 0 and 1. We desire to give high values to terms 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(m) is defined as: $1/(1 + \exp(0.6 \times (m \times 10.5 - 5)))$, where m is defined as: $\mathrm{MAX}(P(a), P(b))$. 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 m 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 terms. For instance, for $a =$ 'ann' and $b =$ 'perceptron', m is 0.2 and SP(m) is 0.85, but for $a =$ 'machin' and $b =$ 'ann', m is 0.6 and SP(m) is 0.31.

Our correlation function AEMI-SP is defined as: AEMI $\times$ 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.

4.2.3 *Other correlation functions*

We also investigated other existing correlation functions. The *Jaccard* function [21] is defined as: $\frac{P(a,b)}{P(a \cup b)}$. When a term describes a more general topic, we expect it to occur quite often and appear with different, more specific terms. Hence, we desire general ("connecting") terms 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 Fig. 2, 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 [28] can be defined as $\mathrm{MIN}(P(a|b), P(b|a))$. The idea is that if we assign the same correlation value to connected terms and connecting terms, 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. However, due to the dominance of 'ai' over 'machine' and 'searching', MIN(P('ai'|'machine'), P('machine'|'ai')) may tend to have higher value than MIN(P('machine'|'searching'), P('searching'|'machine')), which hinders 'ai' from being removed. Alternatively, the MAX function, $\mathrm{MAX}(P(a|b), P(b|a))$, 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.

4.3 Threshold-finding methods

Instead of using a fixed user-provided threshold (as in STC [28] to differentiate strong from weak correlation values between a pair of terms, we examine methods that dynamically determine a reasonable threshold value. Weights with a weak correlation are removed from `CorrelationMatrix` and child clusters are identified.

4.3.1 *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 terms 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

« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 9»

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: $\sum_{i,j} |\mathrm{freq}_i - \mathrm{freq}_j|$, 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 Fig. 5: one from Region 0 through 3, (quality is 17), another one from Region 3 through 5, (quality is 14), and the last one 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 Fig. 6. To identify the threshold inside the bottom region,

Table 4 Distribution of frequency and number of children

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

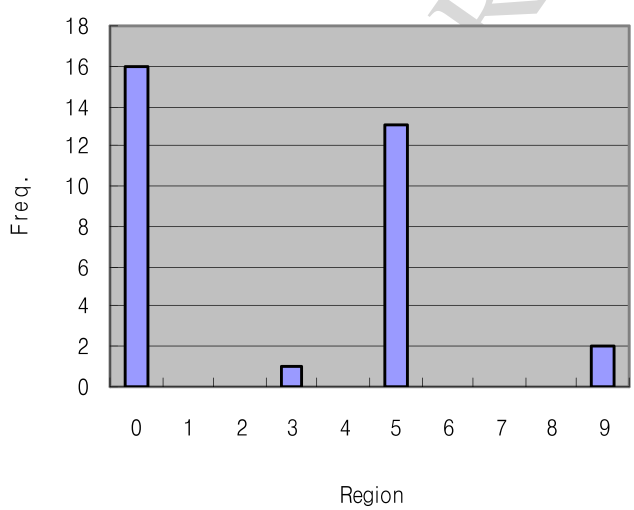


Fig. 5 Shown in a Histogram

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 0.28 and 0.30 can be the threshold. If the bottom does not have zero frequency, we recursively divide the bottom until the frequency is zero.

4.3.2 *MaxChildren*

The MaxChildren method selects a threshold such that maximum of child clusters are generated and the resulting tree is shorter. 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 Fig. 2 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) of the Range column 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.

4.3.3 *Other threshold-finding methods*

There are some other threshold-finding methods that we initially studied, but we found them to be inferior to Valley or MaxChildren, and subsequently they are not included in

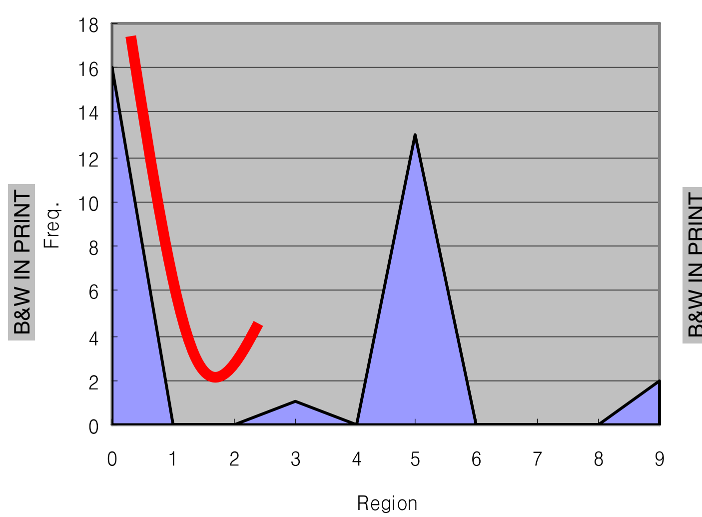


Fig. 6 Find the widest and deepest valley

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 10»

this paper. LargestGap sorts the values and splits them 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.

4.4 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 terms for consideration of co-occurrence. We have been using the entire document length as the window size to simplify our discussion. However, considering two terms 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 terms can represent a concept that is sufficiently specific.

5 Experiments

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

5.1 Evaluation 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 accessed our sever at least 50 times in the first two months and also in the following 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 [4]. This process yielded 13 different users and collected the web pages they visited. The total number of pages that we used is around 1400 and most of the pages contain mostly regular text. The average number of words on a web pages is 1918, with a minimum of 340, and a maximum of 3708.

To find phrases, we used the variable-length phrase-finding (VPF) algorithm [12] because it finds more meaningful phrases than other methods [1, 4]. Phrases are used to enhance the representation of a UIH. 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 authors' bias, we randomly reordered whole clusters from all approaches before we evaluated each cluster.

5.2 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) [22]. 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 [10]. So, we checked two properties of the leaf clusters: the existence of related terms, and possibility of combining terms. For instance, for related terms, 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 terms. Since the terms, 'formal', 'compil', 'graphic', and 'mathemat', are classified as course titles related to the computer science major, this cluster is evaluated as 'good'. For the possibility of combining terms, consider 'research', 'activ', 'class', and 'web' to be in a cluster. In this case, the meaning of the cluster can be estimated as 'research activity' or 'research class' [29], so we regard this cluster as 'good'. A cluster is marked as 'good' when it has more than $2/5$ of the terms 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; however, since 'test info' does not have any conceptual meaning in our opinion, we did not count that phrase. If a cluster contains less then 15 terms and does not satisfy the criteria for 'good' cluster, it is marked as 'bad'. A cluster is marked as

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 11»

'other' when a leaf cluster has more than 15 terms 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 70% of the terms in each child cluster have similar meaning. For example, Cluster 6 and 7 in Fig. 9 contains terms corresponding to course titles in computer sciences and they are siblings: Cluster 6 = {data structure, software engineering, network} and Cluster 7 = {artificial intelligence, database, graphics, and discrete mathematics}. 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.

6 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.

6.1 Building UIH with only words as features

6.1.1 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 [14].

Both methods generated trees whose shapes were mostly 'medium'. For U8, AEMI generated a conceptually related tree as shown in Fig. 7. The tree has a node with two child clusters, which contains words from course titles and hence represents the concepts of different courses (in the dashed box). Cluster 1 represents the homepage of the Computer Science Department. Cluster 3 illustrates academic degree programs. Cluster 4 contains names of faculty members. For U2 with AEMI-SP, the generated tree was 'fat' and had an ABF value of 10.

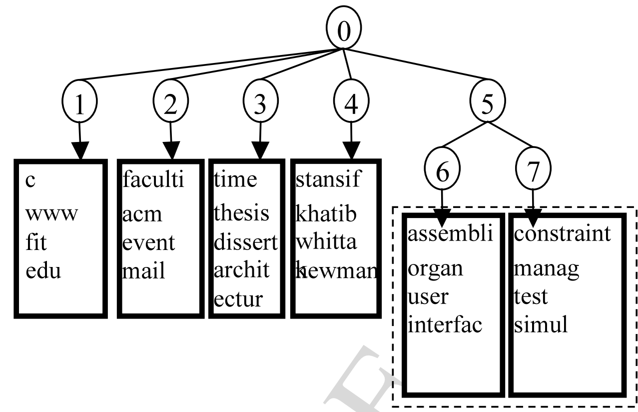


Fig. 7 An example of a conceptual tree of U8 in Table 5

6.1.2 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.

6.1.3 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 (UIH) 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 (clusters): the root and one leaf. These results indicate the differences are not significant.

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 12»

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	

6.2 Building UIH with words and phrases as features

If we can add phrases as a 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. 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 Fig. 8 and Fig. 9. The one from only words has three 'good' leaf clusters (1, 3, and 4) and one 'bad' leaf cluster (5). Cluster 1 shows "research activity" and "research class". Cluster 0 denotes root nodes, which has all words or phrases. The right one which is learned from both words and phrases has all

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

« APIN 10489 layout: Large reference style: basic file: apin56.tex (DL) aid: 56 doctopic: OriginalPaper class: spr-twocol-v1 v.2007/05/18 Prn:21/05/2007; 11:28 p. 13»

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	

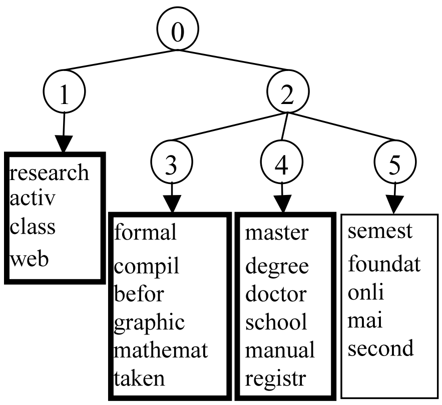


Fig. 8 UIH with words

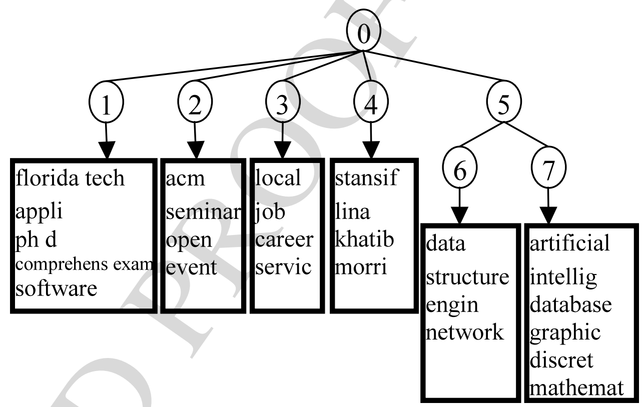


Fig. 9 UIH with words and phrases

'good' clusters; furthermore, it is more descriptive because Clusters 6 and 7 in Fig. 9 contain terms corresponding to course titles in computer science while Cluster 3 in Fig. 8 alone describes course titles. We can say Clusters 6 and 7 in Fig. 9 are conceptually related because both are course titles. We cannot explain why some specific interests in one UIH do not exist in the other UIH. For example, Cluster 4 in Fig. 8 shows that the user (U1) is interested in a Master's or Doctoral degree program, but the interest in the Master's degree does not exist in the UIH in Fig. 9. Cluster 4 in Fig. 9 contains names of faculty members, but they do not appear in the UIH in Fig. 8. Though the difference between the results is not significant, results with phrases achieved higher performance on average.

7 Concluding remarks

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. We used bookmarks for the set of web pages. The bookmarks were assumed to be updated as the user interests change. The UIH should get updated by recalculating bookmarks once in a while. Instead of using the bookmarks, however, the interesting web pages can be collected by other implicit user interest detection techniques. 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 (1400 web pages) on our web server. We also introduced correlation functions and threshold-finding methods for the clustering algorithm. Our empirical results suggested that the AEMI correlation function and the MaxChildren threshold-finding method yielded more meaningful leaf clusters. In addition, by using phrases found by VPF algorithm [12], 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. We faced several problems in evolving this approach such as finding the threshold of DHC and window size. The most difficult part was finding the threshold automatically. We had applied several methods examining the data carefully.

Currently, we are investigating how to apply the generated UIH's to improve the results returned by Google [9]. Based on a user's UIH, pages returned by Google are scored and reranked. For each term that appears in a page as well as

Journal ID: 10489, Article ID: 56, Date: 2007-05-21, Proof No: 1

in the UIH, we use the tree level in the UIH, the number of words in the term, the frequency of the term in the page, and the emphasis of the term in the page (whether a term is in the title, bold, or italic) to calculate the term score. The *personalized* score of a web page is the sum of the term scores. The experimental results in [13] indicate that the *personalized* ranking methods, when used with a popular search engine, can yield more relevant web pages for individual users. The precision/recall analysis shows that our weighted term scoring function can provide more accurate ranking for potentially interesting web pages than Google on average.

Acknowledgements We thank the members of the Laboratory for Learning Research (LLR, http://www.cs.fit.edu/~pkc/llr/) for their comments.

References

1. Ahonen H, Heinonen O, Klemettinen M, Verkamo AI (1998) Applying data mining techniques for descriptive phrase extraction in digital document collections. In: Proceedings of the advances in digital libraries conference, pp 2–11
2. Bellegarda JR (1998) Exploiting both local and global constraints for multi-span statistical language modeling. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing, vol 2, pp 677–680
3. Billsus D, Pazzani MJ (1999) A hybrid user model for news story classification. In: Proceedings of the 7th international conference on user modeling. Springer, New York, pp 99–108
4. Chan PK (1999) A non-invasive learning approach to building web user profiles. In: KDD workshop on web usage analysis and user profiling, pp 7–12
5. Cheeseman P, Stutz J (1996) Bayesian classification (AutoClass): theory and results. In: Advances in knowledge discovery and data mining. AAAI/MIT, Menlo Park, pp 153–180
6. Croft WB, Turtle HR, Lewis DD (1991) The use of phrases and structure queries in information retrieval. In: Proceedings of the SIGIR conference on research and development in information retrieval, pp 32–45
7. Fagan JL (1987) Automatic phrase indexing for document retrieval. In: Proceedings of the 10th annual ACM SIGIR conference on research & development in information retrieval, pp 91–101
8. Fisher DH (1987) Knowledge acquisition via incremental conceptual clustering. Mach Learn 2:139–172
9. Google (2004) http://www.google.com/
10. Han J (ed) (2001) Data mining: concepts and techniques. Kaufmann, San Francisco, p 338
11. Kim H, Chan PK (2003) Learning implicit user interest hierarchy for context in personalization. In: Proceedings of the international conference on intelligent user interfaces. ACM, New York, pp 101–108
12. Kim H, Chan PK (2004) Identifying variable-length meaningful phrases with correlation functions. In: Proceedings of the international conference on tools with artificial intelligence (ICTAI). IEEE, New York, pp 30–38
13. Kim H, Chan PK (2005) Personalized ranking of search results with implicitly learned user interest hierarchies. In: Proceedings of the 11th international conference on knowledge discovery and data mining (ACM SIGKDD WebKDD) workshop on knowledge discovery in the web, Chicago, IL. ACM, New York
14. Lind DA, Marchal WG, Mason RD (2002) Statistical techniques in business & economics, 11th edn. McGraw–Hill, Irwin, pp 377–412
15. Milligan GW, Cooper MC (1985) An examination of procedures for detecting the number of clusters in a data set. Psychometrika 50:159
16. Mitchell T (1997) Machine learning. McGraw–Hill, New York, pp 81–126 and 154–199
17. Mobasher B, Cooley R, Srivastave J (1999) Creating adaptive web sites through usage-based clustering of URLs. In: Proceedings of the 1999 IEEE knowledge and data engineering exchange workshop, pp 19–25
18. Pazzani M, Billsus D (1997) Learning and revising user profiles: the identification of interesting Web sites. Mach Learn 27(3):313–331
19. Pazzani M, Muramatsu J, Billsus D (1996) Syskill & Webert: identifying interesting web sites. In: Proceedings of the national conference on artificial intelligence, pp 54–61
20. Perkowitz M, Etzioni O (2000) Towards adaptive web sites: conceptual framework and case study. Artif Intel 118:245–275
21. Rasmussen E (1992) Clustering algorithms. In: Frakes WB, Baeza-Yates R (eds) Information retrieval: data structures and algorithms. Prentice–Hall, Englewood Cliffs
22. Russell S, Norvig P (eds) (1995) Artificial intelligence: a modern approach. Prentice–Hall, New York, p 74
23. Salton G (1989) Automatic text processing. Addison–Wesley, Reading
24. Trajkova J, Gauch S (2004) Improving ontology-based user profiles. In: Proceedings of the RIAO, Vaucluse, France, pp 380–389
25. Turpin A, Moffat A (1999) Statistical phrases for vector-space information retrieval. In: Proceedings of the SIGIR, pp 309–310
26. Voorhees EM (1986) Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. Inf Process Manag 22(6):465–476
27. Wu H, Gunopulos D (2002) Evaluating the utility of statistical phrases and latent semantic indexing for text classification. In: Proceedings of IEEE international conference on data mining, pp 713–716
28. Zamir O, Etzioni O (1998) Web document clustering: a feasibility demonstration. In: Proceedings of the SIGIR conference on research and development in information retrieval, pp 46–54
29. Zamir O, Etzioni O (1999) Groper: a dynamic clustering interface to web search results. Comput Netw 31:1361–1374