Discovering user communities on the Internet using unsupervised machine learning techniques

G. Paliouras[a,*], C. Papatheodorou[b], V. Karkaletsis[a], C.D. Spyropoulos[a]

[a] *Institute of Informatics and Telecommunications, National Centre for Scientific Research (NCSR) "Demokritos", 15310, Ag. Paraskevi, GREECE*

[b] *Department of Archive and Library Science, Ionian University, 49100, Corfu, GREECE*

Abstract

Interest in the analysis of user behaviour on the Internet has been increasing rapidly, especially since the advent of electronic commerce. In this context, we argue here for the usefulness of constructing communities of users with common behaviour, making use of machine learning techniques. In particular, we assume that the users of any service on the Internet constitute a large community and we aim to construct smaller communities of users with common characteristics. The paper presents the results of three case studies for three different types of Internet service: a digital library, an information broker and a Web site. Particular attention is paid on the different types of information access involved in the three case studies: query-based information retrieval, profile-based information filtering and Web-site navigation. Each type of access imposes different constraints on the representation of the learning task. Two different unsupervised learning methods are evaluated: conceptual clustering and cluster mining. One of our main concerns is the construction of meaningful communities that can be used for improving information access on the Internet. Analysis of the results in the three case studies brings to surface some of the important properties of the task, suggesting the feasibility of a common methodology for the three different types of information access on the Internet.

Keywords: user communities, collaborative filtering, user modelling, machine learning, Web mining

1. Introduction

The separation of "wheat from hay" on the Internet is becoming increasingly important as the amount of information available electronically becomes unmanageable for its non-expert receivers. As a result, a number of service providers have appeared recently in the market to help Internet users separate the information they need out of the plethora of information on the net. Furthermore, new processes such as *electronic customer relationship management* (e-CRM), *Web usage analysis* and *Web mining* have been developed in the last few years. All of these share the same goal: understanding the needs, interests and knowledge of the users of Web sites. The motivation stems from the fact that added value is not gained merely through larger quantities of data on a site, but through easier access to the required information at the right time and in the most suitable form. Seen from a different viewpoint: "*The quantity of people visiting your site is less important than the quality of their experience*" (Schwartz, 1997).

In this paper we examine the exploitation of user modelling techniques for the customisation of information services to the needs of the users. We study three different types of access to information, covering the majority of today's information services on the Internet: query-based information retrieval, profile-based information filtering and Web-site navigation. Our goal is to show how machine learning techniques can be used for automating the construction of user models, providing

* Corresponding author. Tel. +30-10-6503197. Fax. +30-10-6532175

E-mail addresses: paliourg@iit.demokritos.gr (G. Paliouras), papatheodor@lib.demokritos.gr (C. Papatheodorou), vangelis@iit.demokritos.gr (V. Karkaletsis), costass@iit.dmokritos.gr (C.D. Spyropoulos).

useful information for the customisation of Internet services. In particular, we are interested in constructing community models, which represent patterns of usage of the service and can be associated with different types of user. A community corresponds to a group of users who exhibit common behaviour in their interaction with the system (Orwant, 1995). This type of user model has only recently been paid some attention, primarily in the context of collaborative filtering, which aims to personalise a Web service without having to analyse its content. In contrast to earlier, memory-based approaches to collaborative filtering (e.g. Resnick and Varian, 1997), the new approach that makes use of community models – also known as model-based collaborative filtering (e.g. Breese et al, 1998) – organises the users into groups, rather than simply recording information about them individually. In this view, the construction of communities is a more active approach to user modelling than memory-based collaborative filtering, in the sense that models for groups of users are actually constructed.

Through the discovery of communities, an Internet service, such as an e-commerce site, can be treated as a simple community-based system. From that point of view, the users of a service make up a large community, within which smaller communities of users with common characteristics can be identified. These characteristics are usually measurable parameters of their interaction with the system, such as the news articles that they view or the books that they buy, but they can also be augmented with information obtained explicitly from the users, such as their age or their knowledge level. In the latter case, where the users provide explicitly information about them, the user models are often called stereotypes, rather than communities.

The discovery of user communities can facilitate the interaction of humans with a computer system, such as an Internet service, in many ways:

- *Service optimisation*. It provides insight to service providers, helping them to re-organise the system, in order to make it more suitable to the needs of different types of user.
- *Service personalization*. It can help the users identify information of interest to them, in the collaborative filtering fashion mentioned above.
- *Interaction support*. Given the appropriate infrastructure that protects the users' privacy, the service can support interaction among community members. In that case, the acquaintance models (Mamdani et al. 1999) of individual users can be based on the models of the communities in which they participate.

However, the nature and composition of user communities in a highly dynamic environment, such as the Internet, is bound to change continuously. Thus, it is difficult to construct and maintain the communities manually. For this reason, we propose the use of unsupervised machine learning to help in the discovery and maintenance of user communities.

The rest of this paper is structured as follows. Section 2 of the paper provides an overview of related work in user modelling and in particular the use of learning techniques in user modelling. Section 3 presents a detailed description of the proposed approach and introduces the two learning algorithms. Sections 4 to 6, present the results of the three case studies. In all three cases, we have applied both the conceptual clustering and cluster mining techniques. Finally, in section 7, we discuss the relative merits and disadvantages of the two techniques as they arise in the results of the three case studies.

2. Related work

User modelling technology aims to make information systems user-friendly, by adapting the system to the needs of the individual. With the explosive growth of the Internet and the volume of information published on it, the need for user-adaptive systems that help people locate information of interest to them has become imperative. User modelling technology has been successfully used in a variety of domains related to information access: information retrieval (Brajnik and Tasso, 1994; Brajnik et al., 1987; Kay, 1995), filtering (Balabanovic and Shoham, 1997; Maes 1994; Balabanovic and Shoham, 1995; Chen and Sycara, 1998) and extraction (Benaki et al., 1997), adaptive user interfaces (Langley, 1999; Brusilovsky and Schwartz, 1997; Chin 1989) and adaptive Web sites (Perkowitz and Etzioni, 1998, 1999; Ardissono and Goy, 2000).

A user model primarily contains information that characterise the interaction of the user with the system and other users, if interaction between users is supported. As an example, a user model for a service delivering news articles contains the news-reading preferences of a user. Additionally, a user model may contain personal information about the user, such as age, occupation, etc. Further to the

models for individual users, generic user models that apply to groups of users are also widely used. The simplest type of a generic model is a community, which is the subject of this paper and does not presuppose the explicit provision of personal information by the users. If personal information is available, community models are often called stereotypes. A stereotype is one of the earliest types of a generic model that appeared in the literature (Rich, 1983) and is nowadays often encountered in digital libraries and museums (e.g. Esposito et al., 1998; Paternò et al. 1999). Stereotypes are typically based on external knowledge about the user, such as the user's level of expertise or other personal information. For example, a stereotype might link users of a certain age with a specific news category. However, the collection of personal information is a difficult, inaccurate and often undesirable process. It imposes a burden on the user, who needs to provide the information and it may be violating the user's privacy. For these reasons, we focus here on user communities, instead of stereotypes. However, the methods presented here can easily be extended to take personal information into account and thus construct user stereotypes.

Machine learning methods have been applied to user modelling tasks, mainly for acquiring models of individual users interacting with an information system (e.g. Bloedorn et al., 1996; Raskutti and Beitz, 1996; Resnick and Varian, 1997). In such situations, the use of the system by an individual is monitored and the collected data are used to construct the model of the user, i.e., the user's individual characteristics. Such techniques have been used in many agent-based and multi-agent systems, which aim to discover and recommend information on the Internet, like FAB (Balabanovic and Shoham, 1997), Syskill & Webert (Pazzani and Billsus, 1997), WebWatcher (Joachims et al., 1997) and Amalthea (Moukas, 1997). Furthermore, machine learning techniques for user modelling have been used in digital library services, like IDL (Esposito et al., 1998), and in news filtering systems, like News Dude (Billsus and Pazanni, 1999; 2000). The main goal of these systems is to learn and revise user profiles that help in proposing information on a given topic that would be of interest to the user.

In this paper we are concerned with a higher level of generalisation of the users' behaviour: the construction of user communities. If we assume the existence of individual models for the users of the system, which might be constructed by a machine learning method, user communities can be constructed by applying machine learning techniques to the user models. Fig. 1 illustrates the two different levels of learning. In the lower part of the graph, individual user models are constructed by observing the interaction of the user with the system. At the higher level, the user models are used to build communities. Note that in many cases, the construction of individual user models is not possible, e.g. in e-commerce sites where the users do not identify themselves during their interaction with the system. In such cases, interesting usage patterns, corresponding to community models, can be discovered by applying machine learning directly to the low-level usage data. In this paper, we examine both cases, i.e., with and without the existence of individual user models.

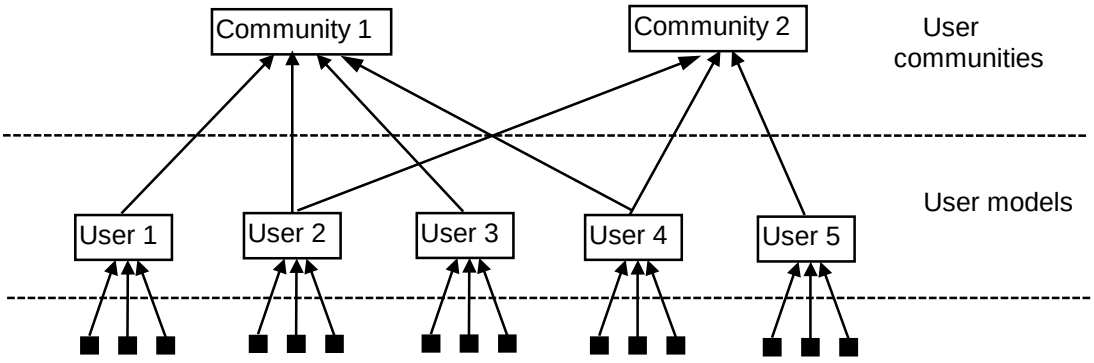

Fig. 1. Two levels of user-model construction.

The construction of user communities can be automated with the use of a learning method, identifying similarities in the interaction of various users with the system. The choice of method depends largely on the type of training data that are available. The main distinction in machine learning research is between supervised and unsupervised learning methods. Supervised learning requires the training data to be preclassified. This usually means that each training item (example) is associated with a unique label, signifying the class in which the item belongs. The important feature of

this approach is that the class descriptions are built relative to the preclassification of the examples in the training set. In contrast, unsupervised learning methods do not require preclassification of the training examples. These methods form clusters of examples, which share common characteristics. When the cohesion of a cluster is high, i.e., the examples in it are very similar, it is labelled as a class.

In the context of constructing user communities, preclassification would mean that each individual in a training set is assigned to a known community. This is an unusual situation, in contrast to the more common goal of wanting to form new communities, by examining the similarities between users. Thus, the construction of user communities is a typical unsupervised learning task. Unsupervised learning tasks have been approached with a variety of methods, ranging from statistical clustering techniques to neural networks and symbolic machine learning. The branch of symbolic machine learning that deals with unsupervised learning is called conceptual clustering and a popular representative of this approach is the algorithm COBWEB (Fisher, 1987), which is used in this paper. One representative of statistical learning is the cluster mining method that was introduced in PageGather (Perkowitz and Etzioni, 1998), a system that associates pages in a Web site according to their use. The second method that we evaluate here is a variant of this method.

The approach to community construction that is presented here is most closely related to a new research area, known as model-based collaborative filtering. In standard, memory-based collaborative filtering systems (Resnick and Varian, 1997; Basu et al., 1998), the individual user models themselves are used to reason about the characteristics of a particular user. This is done in a nearest-neighbour fashion, i.e., the *k* most similar models to the model of a particular user are retrieved and used to recommend extensions to the model of the user that is being examined. This type of reasoning does not involve any learning, in the sense of inferring generic models from the usage data. Such methods, which use solely memory-based approaches, suffer from two problems: they do not scale well to large numbers of users and they do not provide any insight as to the usage patterns that existed in the data. Recently, these problems started to be addressed, by the development of model-based collaborative filtering methods (Breese et al, 1998) and hybrids of model and memory-based methods (Pennock et al, 2000). The models constructed by these methods are community models.

3. Constructing and Evaluating Community Models

3.1. Web usage mining

In principle, the task of constructing community models on the Internet with the use of unsupervised learning is a Web usage mining task, as described for instance in (Cooley, 2000). Communities are built on data collected from the users during their interaction with the system. The goal is to identify interesting behavioural patterns in the collected usage data and base the community models on these patterns. Web usage mining is a natural extension to the various statistical usage figures that system administrators commonly collect. At the level of discovering interesting patterns, usage mining provides more detailed and usable information about the usage of the service than the widely used statistical figures (Paliouras et. al., 1999). However, the discovery of the communities per se does not provide much more information than that provided by memory-based collaborative filtering. In order to turn the patterns discovered with the use of unsupervised learning into actionable knowledge, a further post-processing stage is necessary. The result of this stage is the *model* of each community, i.e., its characteristic features.

The stages of getting from the data to the community models are the same as those of any other data mining task:

Data Collection. During this stage, data from various sources are gathered and their content and structure is identified. Depending on the type of access, the information that is collected varies: queries, profiles or navigation logs. This paper presents in detail the results of three different case studies, where communities are constructed from the three different types of usage data.

Data Pre-processing. This is the stage where data are cleaned from noise, their inconsistencies are resolved, and they are integrated and consolidated, in order to be used as input to the next stage of pattern discovery. In this work, we pay particular attention to the extraction of training data from queries, profiles and navigation logs. In all three cases, the objective is the same: engineering of appropriate features that describe the behaviour of the user and allow the discovery of similarities among different users.

Pattern Discovery. Given the data in the appropriate form, interesting patterns are discovered with the use of machine learning techniques, such as clustering, classification, association rule discovery etc. In this paper we have opted for the use of unsupervised learning, as the most natural candidate for constructing groups of users, based on their similarities. In particular, we compare two unsupervised learning techniques: *conceptual clustering* and *cluster mining*.

Pattern Post-Processing. In this last stage, the patterns are evaluated and translated into actionable knowledge, which usually takes a form that is understandable to humans, e.g. by using reports, or visualization techniques. In our approach we pay substantial attention to the characterisation of communities and the construction of community models. The goal is to construct a prototypical model for each community, which is representative of the participating users and significantly different from the users of other communities.

The following subsections describe the problems encountered at various stages of the usage mining process and the choices that we have made in this paper.

3.2. Data collection and pre-processing

Learning algorithms typically require that the training data be transformed into sets of objects, where each object is described by a set of features. Usage data on the Internet appears in various forms, depending on the type of access that the information service provides. In this work, we examine three different types of service, covering the three most commonly used types of information access. Due to this variety, the translation of the data into a set of objects, maintaining the information that is useful for modelling, is a significant engineering task.

One important issue when pre-processing the data is whether we can identify individual users. This is straightforward when the service has registered users and each interaction of the user with the system is uniquely associated with a particular user code.[1] This is the case in the information filtering case study, where the basic training objects correspond to the models of individual users. However, most Internet-based services are open to everyone and there is usually no mechanism in place for capturing the identity of the user. This is the situation in the other two case studies, i.e., query-based retrieval and Web-site navigation. In those cases, there are no models for individual users and community models are constructed directly from low-level usage data. Learning can then be achieved by changing the definition of the objects in the training set. Since the data cannot be divided on the basis of user codes, other "units of interaction" need to be defined, e.g. access sessions in Web-site navigation. These basic units form the basis for the discovery of similarities.

Another important pre-processing issue is the selection of the feature set, which describes the training objects. Each object is a fixed-length feature vector. Therefore, a limited set of features need to be selected, in order to translate the usage data into a set of objects. The particular choice varies in each case study. In the simple news-filtering case study, where the training objects correspond to user models, each object is simply described in terms of the news categories that the user has selected. In the Web-site navigation case study the training objects correspond to access sessions and we have used two different feature sets: the constituent pages of the session, without any information about the order in which they appear, and the transitions between pages, capturing the sequential nature of the session. Finally, the case of query-based retrieval, where the training objects are user queries, presents an interesting generic problem: the set of keywords that can be used in a free-text query is not restricted. In order to translate such free-text queries into the desired fixed-length format, we made use of language engineering tools and a domain-specific classification hierarchy, i.e., an ontology. With the aid of these resources, we were able to move from a large number of very specialised keywords to few generic terms. The use of such resources like ontology and thesauri can be invaluable when the usage data contains pieces of free text. More generally, external resources and in particular ontologies are often used in user modelling systems. Empirical studies have demonstrated their importance in user model acquisition (Bloedorn et al., 1996; Pazzani and Billsus, 1997). For example OySTER (Müller, 1999), a multi-agent meta-search engine for information filtering, uses a predefined ontology to classify documents (Web pages) selected by the users and then learn user profiles that are expressed in generic ontology terms.

---

[1] Privacy issues arise in the case of registered users, e.g. the administrator cannot disclose personal information about the user, without the user's consent.

3.3. Pattern discovery

Following the procedure presented in 3.2, training data in the form of feature vectors are extracted from the usage data. Each vector is a training object, corresponding to a basic usage entity, i.e., a user model, an access session or a user query. In this study we evaluate two unsupervised learning methods, which process these training data, identifying interesting patterns that correspond to different groups of objects. The two methods are called conceptual clustering and cluster mining and they are presented below.

3.3.1. Conceptual clustering

The branch of symbolic machine learning that deals with unsupervised learning is called conceptual clustering. Conceptual clustering is a type of learning by observation that is particularly suitable for *summarising* and *explaining* data. Summarisation is achieved through the discovery of appropriate clusters, which involves determining useful subsets of an object set. Explanation involves cluster characterisation, i.e., determining a useful description for each cluster. For instance, in the news-filtering case study, summarisation produces groups of similar user models, i.e., the communities, and explanation generates the community models, in terms of news categories.

A popular representative of conceptual clustering is the algorithm COBWEB. It is an incremental algorithm that performs a search to obtain a concept (cluster) hierarchy. The term incremental means that objects are incorporated into the hierarchy as they are observed. The cluster hierarchy that is produced partitions the object space optimally according to a heuristic called category utility (Gluck and Corter, 1985). Category utility is a probabilistic measure of cohesion, i.e., the quality of clusters. It is a trade-off between intra-cluster similarity, i.e., how similar are the objects within a cluster, and inter-cluster dissimilarity, i.e., how dissimilar are different clusters. COBWEB incorporates objects into the hierarchy using the following four clustering operators: placing the object in an existing cluster, creating a new cluster, combining two clusters into a new one (merging) and dividing a cluster (splitting). Given a new object, the algorithm applies each of the previous operators and selects the hierarchy that maximises category utility.

The output of the clustering process is a hierarchy, the leaves of which correspond to single-object clusters, i.e., the individual training objects. Moving from the bottom to the top of the hierarchy, the size of the clusters increases, corresponding to more general concepts. In our case, each cluster corresponds to a community, which is decomposed into sub-communities, down to the level of individual users. Each pair of communities is disjoint, i.e., they do not contain any common users, unless one is a sub-community of the other. This feature is desirable when each user community needs to be treated separately, but becomes restrictive when users naturally belong in more than one community. Another characteristic of COBWEB is that it depends on its incremental character, i.e., it is dependent on the order of the observed objects. Some work has been done in defining alternative category utility measures in order to prevent suboptimal clustering (Fisher, 1996).

In this work, COBWEB was chosen due to its ability to construct probabilistic hierarchies (Thompson and Langley, 1991), which allows the selection of communities at different levels of generality. Furthermore, the symbolic nature of the generated hierarchy facilitates the characterisation of communities, which is necessary for constructing community models. COBWEB has also been used as the basis in the design and evaluation of various other unsupervised learning techniques (e.g. Fisher and Pazzani, 1991; Fisher 1996; Perkowitz and Etzioni, 1999).

3.3.2. Cluster mining

The second unsupervised learning method that we use here is cluster mining (Perkowitz and Etzioni, 1998). Cluster mining discovers patterns of common behaviour, by looking for cliques[2] in a graph corresponding to the users' characteristic features. The algorithm starts by constructing a graph. The vertices of the graph correspond to the users' features, while the edges of the graph correspond to combinations of features as they are observed in the users' interaction with the system. For instance, if the service is a library of computer science articles and the users issue queries about "hardware" and "computing methodologies" an edge is created between the relevant vertices (Fig. 2). The vertices and the edges of the graph are associated with weights, which are computed as the frequencies of the

[2] A clique is a fully connected subgraph, i.e., a set of vertices, which are all connected to each other with edges.

users' choices and their combinations respectively. For instance, in Fig. 2, the category "computing methodologies" seems to be the most popular one among the users of the service and it is often combined with the categories "software", "hardware" and "computing milieux" and less so with the category "mathematics of computing".

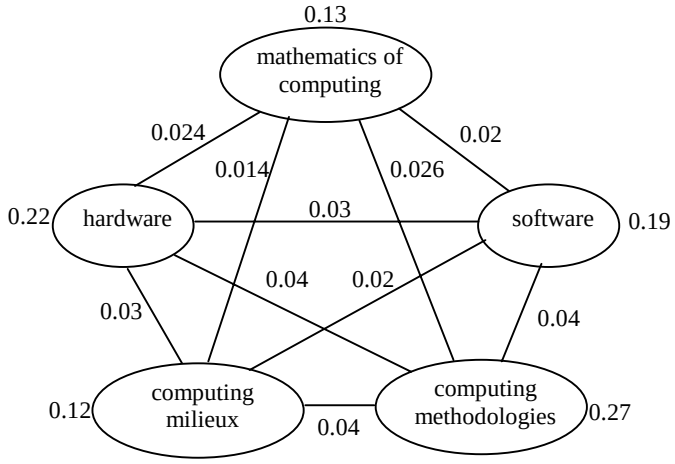

Fig. 2. Non-normalised graph.

One modification that we have made to the basic method is to normalise edge frequencies by dividing them with the maximum of the frequencies of the two vertices that they connect. The effect of the normalisation is to remove the bias for characteristics that appear very often in many users. Using the graph in Fig. 2, the resulting normalised graph is given in Fig. 3.

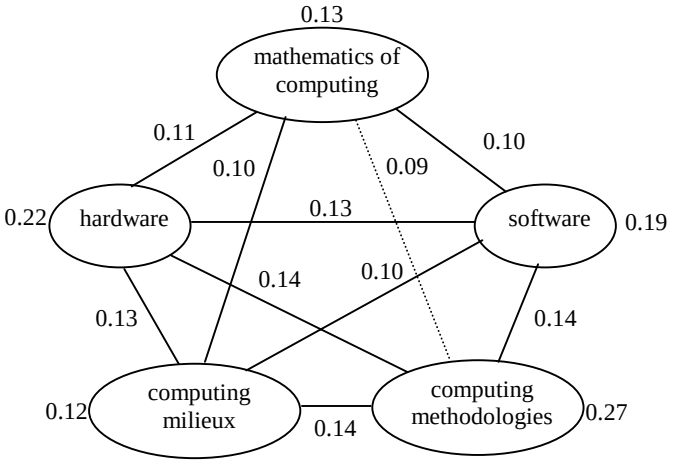

Fig. 3. Normalised graph.

The connectivity of the resulting graph is usually high. For this reason the method requires the use of a threshold, aiming to reduce the number of edges. In our example, if the threshold equals 0.1 the edge between "mathematics of computing" and "computing methodologies" is dropped. The appropriate threshold value differs for different applications and cannot be set to a constant value. In our case studies, we experiment with a range of different threshold values.

One final difference of our implementation of cluster mining, compared to that proposed in (Perkowitz and Etzioni, 1998), is that we do not restrict the size of the cliques, but we generate all maximal cliques[3] of the graph. Despite the large theoretical complexity of the clique-finding problem, in practice the algorithm that we have implemented (Bron and Kerbosch, 1973) is fast.[4] The efficiency of the algorithm allowed a full investigation of the effect of the connectivity threshold.

---

[3] Maximal cliques are cliques that are not subgraphs of other cliques.

[4] It generates all cliques (approx. 200) of a large graph (239 vertices), with an average clique size of 100 vertices, in about 5 minutes on a common SparcServer.

3.4. Pattern post-processing and evaluation

3.4.1. Constructing community models

The final goal of our approach is a set of models, which correspond to behavioural patterns for different types of user. Clustering the users into communities with common behavioural characteristics is a first step towards this goal, but does not provide the desired models. In order to obtain these models we need to identify the descriptive characteristics of each community. The way in which this is achieved differs for different clustering methods, but the underlying ideas are common:

1. *A community model can be expressed in terms of the same features as the underlying usage data*. For instance, if the usage data simply record the pages in a site that are visited by a user, the communities will also be described in terms of the pages in the site. In a more formal language, given a feature set A that describes the objects in the data set, the model of a community C_j consists of a subset of A, A_j, which characterizes the members of the community, i.e., which are usually present in the models of the community members.
2. The selection of the descriptive features is done with the aid of simple metrics, which are based on the idea that *a feature is characteristic of a community if its frequency within the community is significantly higher than its frequency in the whole data set*. The natural choice of metric differs for different clustering methods. For the conceptual clustering algorithm, we have used a squared-difference measure, called FI (frequency increase) (Paliouras et al., 1998), motivated by the *category utility* search heuristic that these algorithms use. Given a feature c, with default frequency f_c, if the frequency of this feature within a community j is f_j, the metric is defined as a simple difference of the squares of the two frequencies: $FI = f_j^2 - f_c^2$. The definition of a representative feature for a community, is simply that $FI > \alpha$, where α is the required level of frequency increase.

In contrast to conceptual clustering, cluster mining does not attempt to form independent user groups. The clusters generated by this method associate characteristic features of the users directly. In this manner, there is no need for an additional stage of community characterisation, as for the conceptual clustering approach. Each clique identified by the cluster mining algorithm is a behavioural pattern. If needed, a user can be associated with the clique(s) that best match(es) the user's individual model. This is not attempted here, as the focus of our work is on the discovery of the behavioural patterns, rather than the user groups themselves.

3.4.2. Evaluation criteria

Having outlined the method to obtain the community models, we need to decide on the desired properties of these models. Our primary objective is to provide useful community models. In order for the models to be useful to humans (e.g. users or service providers), they need to be relatively few in number and small in size. As a result, the *number of models* and their *average size* are two important measurable criteria for the success of a method. The exact figures for these criteria depend on the nature of the problem, e.g. the size of the feature set.

However, a digestible set of models is not necessarily interesting. When there are only small differences between the models, accounting for variants of the same community, the segmentation of users into communities is not interesting. Thus, we are interested in community models that are as distinct from each other as possible, i.e., they share as few features as possible. We measure the *distinctiveness* of a set of community models M by the ratio between the number of distinct features that are covered by these models to the size of the model set M. In other words, we count the number of distinct features that appear in at least one community model and divide that by the sum of the sizes of all models, where the size of each model corresponds to the number of features that it contains. More formally, if there are J communities in M and A_j the features used in the j-th model, distinctiveness is given by the following equation:

$$Distinctiveness(M) = \frac{\left|\bigcup_j A_j\right|}{\sum_j |A_j|} \qquad (1)$$

In the best case, where the community models have no common feature, *Distinctiveness* takes the value 1, as the numerator becomes the same as the denominator. In the worst case, where all models are identical, *Distinctiveness* becomes $1/|J|$, where J the set of community models that have been discovered.

Since we are interested in a small number of models that are distinct, the empty model set trivially satisfies our criteria. In a more realistic situation, we might have a small set of distinct models, which account for only a small part of the usage of the system. In order to avoid this problem, we introduce a further criterion that counterbalances distinctiveness. The new criterion is the overall *coverage* of the community models, i.e., the proportion of features covered by the models. Using the same notation as above, the coverage of the set of models *M* is:

$$Coverage\ (M) = \frac{\left|\bigcup_j A_j\right|}{|A|} \qquad (2)$$

The difference of equation (2) to equation (1) is in the denominator, which now measures simply the total number of features that are used to describe the objects. When the numerator becomes equal to this number, *Coverage* takes its maximum value 1, while the minimum value 0 corresponds to the case where no community models are discovered.

The simultaneous optimisation of distinctiveness and coverage by a set of community models indicates the presence of useful knowledge in the set. It should be stressed that these evaluation metrics are independent of the biases of the community construction methods and thus constitute objective criteria for the evaluation of the methods.

4. Case study A: query-based information retrieval

4.1. Experimental setting

In the first case study we examined an information retrieval service, using free-text queries. The service that we have looked at is the Networked Computer Science Technical Reports Library (NCSTRL). NCSTRL (http://www.ncstrl.org/) is an international collection of computer science technical reports from University departments and research laboratories. In this Web-based digital library, a user inserts keywords, which correspond to titles, abstracts and author names of the technical reports, in a "simple" or a "fielded" form. The system returns the technical reports, which match the keywords, sorted by the providing institutions. The dataset for the experiment contained more than 5,000 user queries stored in the logs of two NCSTRL nodes: NCSR "Demokritos" (Athens, Greece) and FORTH (Heraklion, Crete, Greece).

The aim of our experiment was to identify communities of NCSTRL users, who are interested in similar computer science topics. In other words, we would like to automate the construction of interest groups for the digital library in question. One of the problems that arise when processing free-text retrieval data is the variety of terms that are being used in the queries. The frequency of term co-occurrence in this type of query is too low to allow the discovery of significant similarities. For this reason, we pre-processed the queries with the aid of language engineering tools, before the execution of the learning algorithms. Since the queries referred to Computer Science topics, we replaced the keywords in each query with the most relevant of the eleven top-level categories of the ACM Computing Classification System (http://www.acm.org/class/). The ACM Computing Classification System is an ontology for Computer Science, built by the Association for Computing Machinery. It is almost a tree – with the exception of some cross-references at the same level of the tree – and consists of 4 levels. Each category (node) in the ontology corresponds to a specific research area of Computer Science, which is described by a short phrase, e.g. "automatic programming" or "integrated circuits". We enriched this procedure using WordNet, a general thesaurus (Miller, 1990), which helped in finding synonyms of the query keywords that did not match the ACM Classification terms.

The pre-processing procedure consisted of the following two stages:

1. *Query tokenisation and lemmatisation*. In this stage, the keywords in each query were separated and substituted by their lemma, e.g. 'working' became 'work'. Function words such as articles, prepositions and conjunctions were dropped. The same transformation was done for the phrases representing ACM categories.
2. *Category Matching*. The query keywords were compared with the terms used in the ACM category descriptions, at all levels of the hierarchy. The keywords that did not match any term in the ACM categories were substituted by their synonyms retrieved from WordNet. If none of the synonyms matched the ACM terms, then this query keyword was dropped. Each surviving query was mapped onto a set of top-level ACM categories, called the *ACM query*, as follows:

a. Each query was compared to ACM category descriptions, at all levels of the hierarchy, and two figures were computed: the *query coverage ratio*, which is the number of query keywords appearing in an ACM description divided by the total number of keywords in the query, and the *ACM category coverage ratio*, which is the number of terms in an ACM category description appearing in a query, divided by the total number of terms in the description.
b. For each query, all ACM categories that matched the query were sorted in descending order of the two ratios, giving priority to query coverage.
c. The first ACM category in each ranked list was chosen and its top-level ancestor in the ACM classification hierarchy was added to the corresponding *ACM query*.
d. The query keywords covered by the *ACM query* were dropped and the process was repeated for the remaining words.
e. The iterative process ended when either all query keywords were dropped or the ranked list of matching ACM categories was exhausted, i.e., the remaining keywords of the query could not be mapped onto the ACM hierarchy.

The above process filtered the queries that did not match the ACM ontology, leading to a dataset of 3,200 queries, containing top-level ACM categories as terms. This dataset contained a substantial amount of duplication, as there were just 187 unique queries. Thus, with the use of the ACM hierarchy, we have achieved a first level of clustering through the generalisation of the query keywords.

Finally, the queries were translated into training objects in the feature-vector representation that the learning algorithms use. Each query became a binary feature vector. Each bit of the vector corresponded to one of the 11 top-level ACM categories. It was "on" when the category appeared as a query term and "off" otherwise. The conceptual clustering algorithm constructed groups of similar vectors, i.e., queries with common terms. The cluster mining algorithm, on the other hand, identified cliques in the 11-node graph, which represented the co-occurrence of terms in the queries. The experimental results for each of the algorithms are presented below.

4.2. Number and size of the community models

In this case study, COBWEB generated a hierarchy consisting of 3,107 non-leaf nodes, corresponding to groups and subgroups of user queries. As mentioned above, the 3,200 user queries have been clustered to only 187 unique ones with the use of the ACM classification hierarchy. For this reason, there are only 111 nodes in the COBWEB hierarchy that have a category utility < 1.0. In these 111 "communities" there is still some uncertainty about at least one ACM category, i.e., there are people in the community who are interested and others who are not interested in some of the categories. Each of the remaining 2,996 groups consists of identical queries and is thus of limited interest. Fig. 4 presents the top two levels of the hierarchy. We chose not to examine lower levels, as the category utility scores in the second level are already close to 1.0, i.e., there is already a high degree of agreement among the community members. Thus, level 2 is appropriate for constructing the community models.

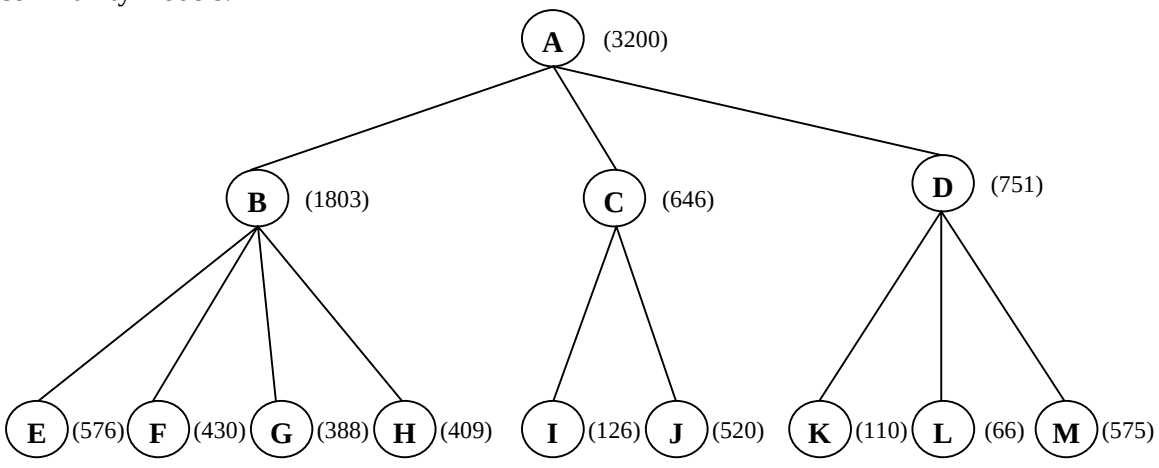

Fig. 4. The top two levels of the hierarchy generated by COBWEB in case study A. Node A is the root of the hierarchy, covering the whole dataset. The numbers in brackets correspond to the number of objects (user queries) covered at each node.

The cluster mining algorithm groups together query terms, i.e., ACM categories, that co-occur frequently. The feature graph in this case study represents the co-occurrence frequency between query terms. The only external parameter in the cluster mining algorithm is the connectivity threshold, which determines whether an edge of the graph will be removed or not. In order to examine the effect of this parameter to the quality of the generated models, the connectivity threshold was varied from 0 to 0.95, at 0.05 intervals. Low values of the threshold allow weaker links between ACM categories to survive, i.e., the graph has more edges. When the threshold is 0, no edge is removed. The threshold has an important effect on the size and the number of cliques that are discovered. The higher the connectivity of the graph, i.e., the larger the number of edges for the same number of vertices, the more likely it is for large cliques to exist in the graph. In the extreme case, where the vertices of the graph are fully disconnected, there are as many cliques as the vertices and they are all singleton (1 vertex per clique). This situation occurs for threshold values greater than 0.15 in this case study. The number of cliques that are discovered for each threshold value also varies with the threshold. For high connectivity of the graph, i.e., low threshold, the size of the cliques is large and therefore there cannot be many cliques in the graph. The number of cliques generally increases as the connectivity of the graph falls.

4.3. Distinctiveness and coverage of the models

As explained in subsection 3.4, a measurable indication that a set of community models is interesting can be obtained by optimising the distinctiveness and the coverage of the models. Since, we are interested in the combined optimisation of distinctiveness and coverage, we would like to present the results along those two dimensions in a combined manner. A good choice for such a presentation is the use of Receiver Operating Characteristic (ROC) curves. ROC curves are commonly used for cost-sensitive classification tasks, such as medical diagnosis, in order to present the trade-off between two types of error, e.g. over-diagnosis and under-diagnosis. The two types of error are measured by two corresponding measures, usually called *sensitivity* and *specificity*. A ROC curve is a plot of sensitivity against the opposite of specificity (1-specificity). Adapting this idea to our measures, we plot coverage against the opposite of distinctiveness (1-distinctiveness). We name this type of plot a *trade-off* curve, in order to avoid confusion with ROC curves, since we are not measuring sensitivity and specificity. The results that we obtained in this first case study are shown in Fig. 5. Each curve is generated by measuring coverage and distinctiveness for different values of the FI (frequency increase) threshold (α) and the connectivity threshold. Similar to the ROC curves, the optimal point of maximum coverage and distinctiveness is the top-left corner of the square, while the area underneath each curve is a measure of the overall quality of the corresponding method.

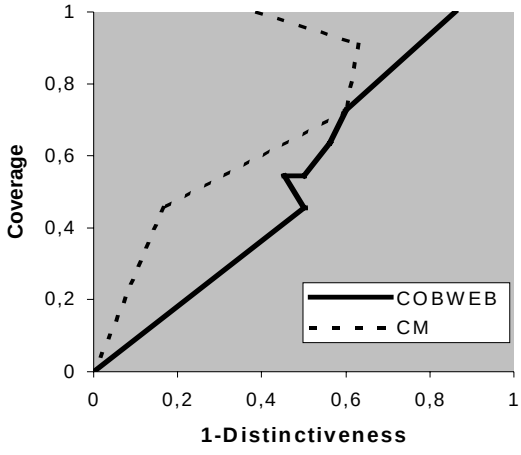

Fig. 5. Trade-off curves for case study A (CM=Cluster Mining).

The results presented in Fig. 5 indicate that the cluster mining method is doing consistently better than COBWEB, in terms of coverage and distinctiveness. Particularly interesting is the relatively high value of distinctiveness, 0.61, when the coverage reaches its maximum. At this threshold level, no edge of the feature graph is removed and only two communities are generated. Despite their fairly large size, the two community models are very different, in terms of the features that they contain. COBWEB, on the other hand, is only able to reach full coverage at the expense of very low distinctiveness. A similar difference between the two methods is observed when looking at the best

result in terms of distinctiveness (excluding the point of origin). COBWEB does not reach a level of distinctiveness above 0.5, while cluster mining reaches 0.83 at roughly the same level of coverage as COBWEB.

4.4. Indicative community models

In order to illustrate the type of information that the community models provide, we examine here an indicative model for each of the two methods. Table 1 presents the community models generated by COBWEB when the FI threshold is set to 0.15. The results reveal three dominating trends, corresponding to the three communities at the first level of the COBWEB hierarchy (see Fig. 4). The three communities could be labelled "Computer Systems" (node **B** in Fig. 4), "Information Systems" (node **C** in Fig. 4) and "Computing Methodologies" (node **D** in Fig. 4). In each of these three broad communities there is further sub-categorisation, combining other interests of the users, e.g. "Hardware" and "Software". Community **E** serves as a "miscellaneous" cluster and does not seem to be homogeneous. The main similarity between the queries in this cluster, is that they contain very few keywords, usually just one. Furthermore, each of these keywords does not occur frequently enough to justify the construction of a separate community. The construction of this "miscellaneous" cluster is an interesting result in terms of modelling, because it shows that a subset of the users are better left unclassified. It is important to identify this special cluster and not treat it as a normal cohesive community.

Table 1
Community models generated by COBWEB in case study A. In brackets are the FI values.

Community	Interests
E	
F	Computer Systems Organisation (1.0)
G	Software (1.0)
H	Hardware (1.0)
I	Information Systems (1.0), Computing milieux (0.63), Computing methodologies (0.28)
J	Information Systems (1.0)
K	Computing methodologies (1.0), Hardware (1.0)
L	Computing methodologies (1.0), Software (1.0)
M	Computing methodologies (1.0)

Similarly, Table 2 presents the non-singleton cliques generated by the cluster mining method, when the connectivity threshold takes the value 0.1.

Table 2
The non-singleton cliques generated in case study A. The connectivity threshold is set at 0.1.

Clique	Interests
1	Hardware, Software, Computing Milieux, Computing Methodologies
2	Hardware, Software, Computing Milieux, Mathematics of Computing
3	Hardware, Computer Systems Organisation
4	Theory of Computation, Mathematics of Computing
5	Information Systems, Software, Computing Milieux, Computing Methodologies
6	Information Systems, Software, Computing Milieux, Mathematics of Computing

As expected, there is some agreement between the models generated by cluster mining and those constructed by COBWEB, in terms of the ACM thematic categories. However, the distinction between the three broad trends that was noticed in the COBWEB results, i.e., "Computer Systems", "Information Systems" and "Computing Methodologies", is less clear in the results presented in Table 2. The cliques seem to capture more subtle relationships between thematic categories across the three trends. An example of such a relationship is the third clique, which relates "Hardware" with "Computer Systems Organisation". Another intuitive relationship that did not appear in the COBWEB results is the one between the "Theory of Computation" and "Mathematics of Computing".

Concluding, the above results are suggestive of the usage of NCSTRL, with respect to the ACM thematic classification hierarchy. One could envisage using the results of such an analysis to organise the material in NCSTRL and provide an alternative method of search for computer science technical reports. According to this alternative, the users would be guided by the system in their search, through a "thematic map" of NCSTRL. This "map" does not have to take the hierarchical form of the ACM classification hierarchy, but could be more adapted to the usage analysis results. For instance, different areas of the "map" could be allowed to overlap. Furthermore, the community models could be used to make NCSTRL a community-based system, in which the users could either choose to join different communities or they could be assigned to communities, based on the queries that they usually issue. On that basis, collaborative filtering could also be used to personalize the query results to the interests of each user, according to the communities in which the user belongs.

5. Case study B: profile-based information filtering

5.1. Experimental setting

In the second case study we examined a news-filtering system. This system collects information from various sources on the Internet and forwards it to its users, according to their profiles. The news articles are organised by the information provider into 24 news categories, e.g. "sports", "computers", etc. During their registration, the users specify a subset of these categories, which correspond to their personal interests. This personal list of news categories constitutes the user profile, which determines what news the user receives. The profile can be modified by the user at any point in time, reflecting changes of interest.

The dataset for the experiment contained 1,078 user profiles, with an average of 5.4 news categories specified in each profile. These profiles were used as the training set for the two learning methods. Each profile was translated into a binary feature vector, specifying the news categories that the user was interested in. As in the first case study, each bit in the vector corresponded to a category and was "on" if the user was interested in this category and "off" otherwise. COBWEB grouped the users according to their interests, as expressed in their profiles. The cluster mining algorithm identified cliques in the 24-node graph, which represented the co-occurrence of news categories in the user profiles. Thus, the cliques represented clusters of related news categories according to the users' interests. The experimental results are presented below.

5.2. Number and size of the community models

The hierarchy generated by COBWEB in this experiment consisted of 699 nodes, of which the first three levels are presented in Fig. 6. An interesting property of the tree in Fig. 6 is the balanced split of objects in different branches. The communities at the same level of the tree are of comparable size. Due to the balanced shape of the hierarchy, the choice of a set of communities is not as straightforward as in the first case study. As it can be seen in Fig. 6, levels two and three of the community hierarchy contain a manageably small number of communities of comparable size. For this

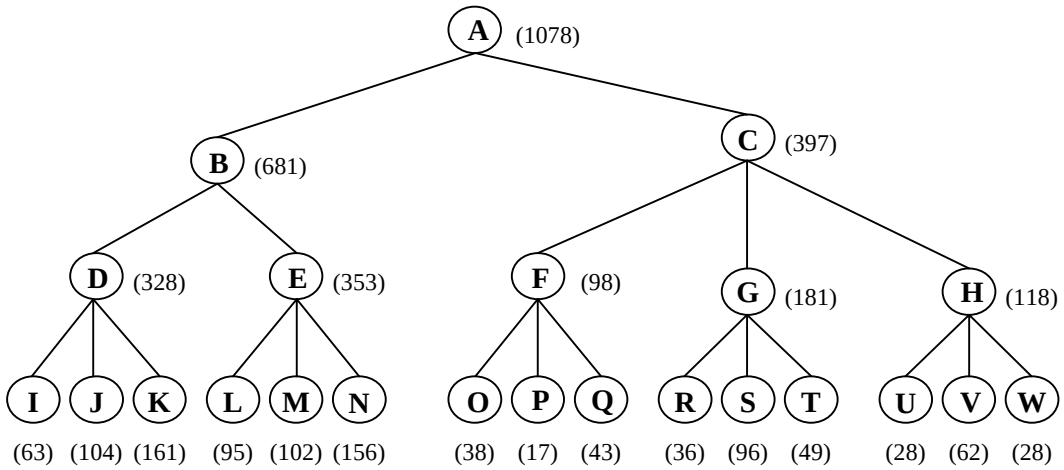

Fig. 6. The top three levels of the hierarchy produced by COBWEB in case study B. The numbers in brackets correspond to the number of objects covered at each node.

reason, we chose to examine both levels of the hierarchy, i.e., nodes {**D, E, F, G, H**} and {**I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W**}, as two separate sets of communities.

The community models generated by cluster mining for different values of the connectivity threshold follow a similar pattern as in case study A. The average size of the discovered cliques falls as the threshold increases and the graph becomes fully disconnected when the threshold is 0.75. The number of cliques increases almost monotonically as the threshold increases and the connectivity of the graph falls.

5.3. Distinctiveness and coverage of the models

As in case study A, we measured the distinctiveness and coverage of the models generated by the two methods for different values of the corresponding thresholds. Fig. 7 presents the trade-off curves for the models generated by COBWEB at level 2 (L2) and level 3 (L3) of the hierarchy, as well as the models constructed by cluster mining.

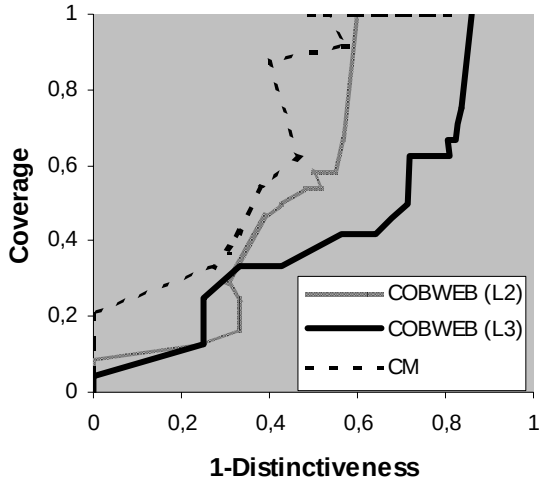

Fig. 7. Trade-off curves for case study B.

Comparing the two levels of the COBWEB hierarchy, it seems that for most values of the FI threshold, level 2 (L2) is preferable to level 3 (L3). At first this observation is surprising, because the coverage of L2 is consistently lower than that of L3, for the same FI threshold. This is due to the fact that the L3 communities are more specialised than the L2, i.e., the frequency increase for the categories in the L3 communities is higher than in the more general L2 communities. Therefore, the fact that the L2 trade-off curve is generally better than that for L3 shows that the distinctiveness in L3 is lower than in L2, at the same level of coverage. This is due to the large number of communities in L3, which are less distinct than those in L2. This situation shows the power of the trade-off curves, which present the interaction between the two measures and facilitate the choice of the best method.

As in the first case study, cluster mining seems to be doing consistently better than COBWEB at both levels of hierarchy. It generates community models with high coverage and high distinctiveness.

5.4. Indicative community models

Similar to case study A, we present here indicative models for the two methods. Table 3 lists the models for the five L2 communities discovered by COBWEB, when the FI threshold is set to 0.5. Communities **E**, **G** and **H** are well-separated, corresponding to one group of people interested in the Internet, a second one interested in economics and a third interested in computers. Group **F**, consists of people interested mainly in economics and finance, but also in computers. Some interest in computers is to be expected from the users of any system on the Internet. Finally, cluster **D** serves as a "miscellaneous" category, covering people who are very specific in their choices, i.e., most categories are not of interest to them.

An interesting property of the models in Table 3 is that a large number of news categories are not covered. In general, these are the categories that are chosen by either too few or too many users and are therefore not useful for characterising the communities. Filtering out these two types of category is a desirable effect. Coverage can increase, by moving selectively to lower levels of the concept hierarchy. For instance, the children of node **H** give meaningful and concise communities,

corresponding to categories that are related to computers, e.g. electronics, networks, telecommunications. However, this is not the case for all of the five communities in Table 3. For instance, the children of node **E** do not provide further meaningful subgroups in the community. The ability to vary the generality level of the communities is an important advantage of the hierarchy generated by COBWEB.

Table 3
Community models on the second level of the COBWEB hierarchy. In brackets are the FI values

Community	Interests
D	
E	Internet (0.55)
F	Economic Indicators (0.73), Economics & Finance (0.68), Computers (0.6), Transport (0.53), Financial Indicators (0.5)
G	Economic Indicators (0.58), Economics & Finance (0.61)
H	Computers (0.53)

Table 4 presents the non-singleton cliques generated by cluster mining, when the value of the connectivity threshold is 0.4. The broad qualitative conclusion from the results in Table 4 is similar to that expressed above for COBWEB: there is a group of people who are mainly interested in economic and financial news (cliques 3, 5 and 6), another group who combine such interests with more technical news related to computers (clique 1) and finally those people who are purely technically oriented (cliques 2 and 4). Despite their similarity, the community models in Tables 3 and 4 are not identical. For instance, cluster mining discovers a strong link between Computers and Telecommunications that was not discovered by COBWEB. In this manner, cluster mining confirms and extends the relationships that were discovered by COBWEB. Interestingly, cluster mining has discovered a further trend corresponding to people who are interested in entertainment electronics and sports (clique 7).

Table 4
The non-singleton cliques generated in case study B, when the connectivity threshold is 0.4

Clique	Interests
1	Telecommunications, Computers, Internet, Industries, Economics/Finance
2	Telecommunications, Computers, Networks
3	Telecommunications, Economic indicators, Economics/Finance
4	Hardware, Software
5	Financial indicators, Economic indicators, Economics/Finance
6	Financial indicators, Economic indicators, Financial markets
7	Sport, Entertainment electronics

Concluding, the community models derived by the two clustering algorithms can be used in several ways to improve the news-filtering service. For instance, based on the assignment of users to communities, the system could make suggestions to the users about news categories that might interest them, but they have not included in their profile. Furthermore, if the users have agreed to share personal information, the system could suggest to each user, other users with similar interests for further communication. Moreover, the service is associated with a Web site providing on-line news. This site is organised according to the 24 news categories that were used here. The results presented above could be used to modify the organisation of the site, tailoring it to the interests of different communities. Such a re-organisation of the site, assumes that the communities constructed using the news-filtering data, apply also to the visitors of the site. This assumption should be verified by looking at the navigational behaviour of the visitors of the site, which is the subject of case study C.

6. Case study C: Web-site navigation

6.1. Experimental setting

In the third case study we examined the construction of user communities from access log data in a Web site. The aim here was to identify patterns in the navigational behaviour of the visitors. For this experiment, we used the access logs of the Advanced Course on Artificial Intelligence (ACAI ’99)

Web site (http://www.iit.demokritos.gr/skel/eetn/acai99/). The log files consisted of almost 5,200 Web-server calls (log file entries) and covered the period of between 7 and 26 of May 1999. Each log entry recorded a visitor's access date and time, computer IP address and domain name, and the target page (URL).

In order to construct a training set for the clustering algorithms, the data in the log files passed through two stages of pre-processing:

1. Access sessions were extracted.
2. The paths recorded in the access sessions were translated into feature vectors.

Extracting access sessions from log files is a complex procedure, in which uncertainty plays a significant role. This process involves the following stages:

1. Grouping the hits by date and IP address.
2. Selecting a time-frame within which two hits from the same IP address can be considered to belong in the same access session.
3. Grouping the pages accessed by the same IP address within the selected time-frame to form a session.

In order to select the time-frame, we plotted the frequency distribution of the page transitions in minutes. According to this distribution, transitions from one page to another, which are done with a time interval longer than one hour, had almost zero frequency. Thus, a sensible definition of the *access session* is a sequence of page transitions for the same IP address, where each transition is done at a time interval smaller than one hour. Based on this definition, our log files consisted of 1,006 access sessions.

Concerning the translation of access sessions to feature vectors, we examined two alternative approaches. In the first approach each feature in the feature vector represented the presence of a particular page of the Web site in the session. There were 41 pages in the site that were visited at least once, and the feature vector consisted of 41 binary features. In the second approach, we used transitions between pages, rather than individual pages as the basic path components. Clearly the number of all possible transitions between 41 pages is prohibitively large. Even the number of different transitions that appear in the log files is very large. Thus, we needed a method to reduce the number of features. This reduction was achieved by examining the frequency distribution of the transitions from one page to another. We decided on a cut-off frequency of 20, which was the point where the distribution was becoming flat. As a result, transitions that occurred fewer than 20 times in the training data were removed from the feature set. Additionally we removed all transitions from a page to itself. As a result, 27 transitions survived this selection and were used to form the binary feature vector.

Thus, in the first representation of the problem (page representation), COBWEB generates communities based on the number of common pages among different sessions. The cluster mining algorithm constructs groups of pages, which often co-occur in access sessions. In the second representation (transition representation), each binary vector corresponds to a sequence of page transitions. Thus, COBWEB generates communities on the basis of common transitions among sessions, while cluster mining identifies clusters of co-occurring page transitions. Some of the desired results in this case study are: paths that are commonly followed by different groups of people, pages that people often visit in the same session through different paths, etc. Such information is valuable for the adaptation of the site to the preferences of the users.

6.2. Number and size of the community models

As explained above, we did two different experiments with two different feature sets. In the first test, COBWEB generated a hierarchy of 1,752 nodes. Starting at the top node and moving down the hierarchy, the set of 1,006 sessions was divided into two subsets, each of which was subdivided into three smaller subsets. Thus, at the second level of the hierarchy there were six clusters. This is a manageable number of clusters to examine the characteristics of the community models.

In the second test, where sessions are represented as sets of transitions, COBWEB generated a hierarchy of 1,888 nodes. Focusing again on the top nodes of the hierarchy, the set of 1,006 sessions was first split into three subsets, which where further subdivided into eight subsets on the second level. We concentrated our analysis on this set of eight clusters.

The cluster mining algorithm was also tested using both representations. An interesting difference of the results in this case study with the results in the previous case studies is that the number of

cliques for both representations is very large for small values of the connectivity threshold. As the connectivity threshold increases, the number of cliques decreases quickly and then increases slowly to reach the terminal situation, where each community corresponds to a single page or a single transition between pages. The large number of cliques for small connectivity thresholds is a result of the high connectivity of the corresponding graphs. For instance, in the case of the page-based representation, when the threshold value is 0 the graph is fully connected and therefore forms a single clique of 41 nodes. As some of the edges are removed, this large clique breaks into a large number of highly-overlapping cliques. When the connectivity decreases further, the cliques become more disjoint and their number decreases. However, it does not go far below the terminal level, where we have only singleton cliques.

6.3. Distinctiveness and coverage of the models

The aim of examining the trade-off between coverage and distinctiveness in this case study is two-fold: to compare the two learning methods, as in the previous two case studies, but also to examine the appropriateness of each of the two representations. Figs. 8 and 9 present the trade-off curves for the two different representations of the problem.

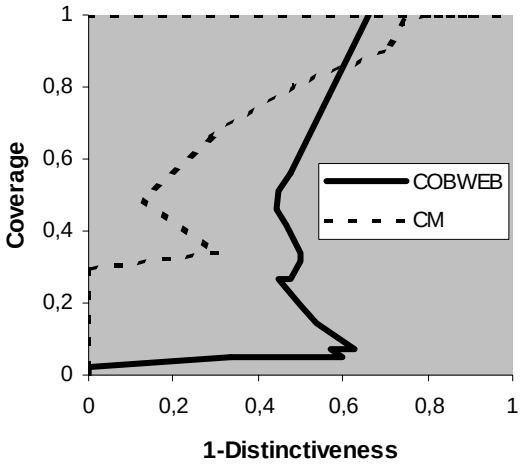

Fig. 8. Trade-off curves for the page representation of the case study C.

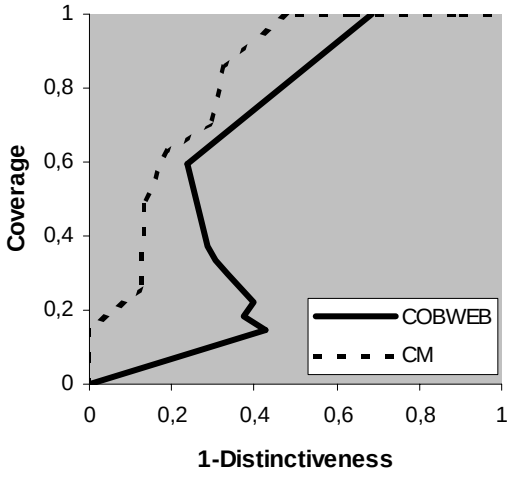

Fig. 9. Trade-off curves for the transition representation of the case study C.

As it was the case in case studies A and B, cluster mining performed generally better than COBWEB, independently of the representation that was used. The results with the transition representation were particularly good, achieving a high level of both coverage and distinctiveness. COBWEB also performed better using this representation, which is an indication that the use of transitions, as the basic building block, allows the construction of interesting navigation patterns for user communities. Some idea of the patterns that were discovered can be gained by examining indicative community models, as explained in the following section.

6.4. Indicative community models

As in the previous two studies, we present here indicative models for the two methods, using the two different representations of the problem. Tables 5 and 6 show the community models discovered by COBWEB, when the FI threshold is set to 0.05. In the second column of Table 5, the pages in each community model are presented in decreasing order of frequency increase, i.e., the most representative pages are near the beginning of the list. Community **D** serves as a filter for the users who access a small number of pages (typically one). There does not seem to be a particular preference for some pages within this community. Community **F** groups a large number of users who visited only the first page of the site. In general the three first sibling communities (**D, E, F**) consist of short sessions. The longer sessions are assigned to the last three communities (**G, H, I**), which are hardly differentiable by the models that are generated. Thus, the main conclusion of this experiment is that looking at the sessions as bags of pages does not help in analysing the navigational behaviour of the users of the site.

As in all of the experiments with COBWEB that have been presented in this paper, the first community in Table 6 is large and does not have a representative model. It clusters together mainly small sessions, including the large number of sessions, in which only the first page of the site was visited. In other words, the communities **D** and **F** in Table 5 are grouped together into one community (**A**) in Table 6. The other community models consist of sequences of page transitions, sorted in decreasing order of frequency increase. The first observation is that there are clear differences between the models. More interestingly, however, the transitions in most of the communities seem to make up paths through the pages. This is not imposed by the representation, which encodes transitions between pairs of pages, rather than complete paths. The representative paths for the communities express very interesting navigational patterns. For instance, the users of the large community **B**, seem in general to follow the path: (1→19→23→24→25). Page 24 contains registration information for ACAI '99 and page 25 is the registration form. Thus, the community consists of visitors who were interested in registering for the course. This observation could be utilized in several ways by the administrator of the site, e.g. by adding a direct link from page 1 to page 25, or by suggesting a transfer to page 25, once the transition from page 1 to page 19 is observed. Thus, the results of COBWEB confirm the suspicion that the transition representation provides interesting behavioural patterns.

Table 5
Community models for the second level of the COBWEB hierarchy, using pages as features

Community	Pages
D	
E	31,1,30
F	1
G	1,22,20,31,27,28,7
H	1,31,22,27,20,2,19,30,9,28,10,23,24,7,15,29,3,14,26,17,11,12,25
I	1,24,19,23,22,25,31,10,30,14

Table 6
Community models for the second level of the COBWEB hierarchy, using transitions as features

Community	Transitions
A	
B	24→25, 23→24, 1→24, 1→19, 19→23
C	1→22, 22→20, 20→31, 31→27, 27→7, 19→23
D	22→31, 1→22
E	22→27, 1→22
F	1→30
G	1→30, 8→1, 1→8
H	30→31, 1→30

Similarly, Tables 7 and 8 present indicative community models discovered by cluster mining for the two representations. The connectivity threshold is 0.35 in the page representation and 0.2 in the transition representation. In the page representation, the results are as inconclusive as those generated using COBWEB. In contrast, using transitions to represent access sessions, the results are much more interesting. The cliques in Table 8 resemble closely the community models for COBWEB in Table 6. COBWEB's community **B** is a combination of two cliques: **1** and **2**. Community **C** is also a combination of three cliques: **4**, **6** and **10**. Communities **D**, **G** and **H** can be associated to cliques **3**, **8** and **5** respectively. This agreement in the results generated by the two different algorithms is a further indication that the transition representation is more appropriate than the page representation for modelling navigational behaviour. The breakdown of community models from Table 6 to multiple cliques in Table 8 shows that the communities generated at the second level of the COBWEB hierarchy are broader than the communities to which the cliques in Table 8 correspond.

Concluding this case study, the above analysis justifies the claim that Web usage mining provides much more actionable knowledge than the simple Web usage statistics that are commonly collected by system administrators. The results obtained by usage analysis can be used to modify the structure of a Web site, in reaction to the interests of the visitors and/or make the site adaptive to different types of visitor. Furthermore, the patterns that we obtained when using transitions between pages as the basic

building block for analysis, seem much more interesting than the patterns discovered by the usual bag-of-pages approach. Higher-order representations, i.e., longer sequences of page transitions, may be interesting, but are likely to increase the dimensionality and reduce drastically the density of the training data. This can have a seriously negative effect on the ability of the learning algorithms to generalise.

Table 7
The non-singleton cliques in case study C, using the page representation. The connectivity threshold is set to 0.35

Clique	Pages
1	22,27,20
2	22,27,31
3	22,19
4	2,9
5	3,26
6	4,5,6,21
7	10,23
8	11,12
9	11,15
10	15,29
11	16,18
12	23,24,19
13	23,24,25
14	28,27
15	28,31
16	37,38

Table 8
The non-singleton cliques in case study C, using the transition representation. The connectivity threshold is set to 0.2

Clique	Transitions
1	1→19, 19→23, 23→24, 24→25
2	1→24, 24→25
3	1→22, 22→31
4	1→22, 22→20
5	1→30, 30→31
6	22→20, 20→31, 31→27
7	22→20, 20→27
8	1→8, 8→1
9	1→9, 9→2
10	20→31, 31→27, 27→7
11	19→23, 23→14
12	23→14, 27→7
13	1→2, 2→11
14	2→11, 11→12
15	1→23, 23→24

7. Conclusions

Efficient and effective access to on-line information becomes increasingly critical as the amount of information that becomes on-line increases at an overwhelming pace. For this reason the number and the variety of services for delivering information over the Internet is continuously increasing. In the three case studies that were presented in this paper we examined three of the most commonly used types of information service: query-based retrieval, profile-based filtering and Web-site navigation. The common objective in all three studies was to analyse the behaviour of the users of the service and to provide useful information to the provider, in order to improve the service, either through its re-organisation or through its personalisation. The approach that we have adopted was to construct user communities, corresponding to groups of users with similar behaviour. The end result was a behavioural pattern for each community, which provides much richer information to the provider than the commonly-used statistical figures about the usage of a service.

In all three case studies the discovered patterns are intuitive, while at the same time they reveal interesting aspects of the users' behaviour. As a result, they constitute actionable knowledge for the service provider, indicating ways to improve the service. For instance, the user-friendliness of NCSTRL could be enhanced by augmenting the keyword-based query mechanism with community-specific thematic maps, based on the ACM classification, as mentioned in section 4.4. Similarly, community models can be used to personalize the news-filtering service examined in section 5 and facilitate interaction between users with common interests. Finally, the organisation of the ACAI'99 Web site could be improved, as indicated by the patterns presented in section 6.4.

One of the important technical issues that have arisen in this work was the necessary engineering of the data collected on the three services. Each of the three studies dealt with a different type of data and used a different method to transform these data into a training set for the learning algorithms. The easiest case was the profile-based filtering service, in which each object was simply the profile of a user. In the case of the query-based information filtering service we employed language engineering tools and a domain-specific classification hierarchy, in order to reduce the dimensionality of the problem and generate the object set. Finally, in the case of the Web site, we developed a procedure for

generating access sessions from individual requests for pages on the site. In this case study, we also examined two different kinds of feature: individual pages and page transitions. The latter seemed to capture more of the navigational behaviour of the user and provided better results. The conclusion of this effort is that data engineering is an important task in the analysis of user behaviour on the Internet. We dealt with three commonly-used types of data and proposed a solution for each.

Another important issue is the choice of the learning method, which will construct the communities. In this work, we have looked at two methods: conceptual clustering and cluster mining. The two methods differ significantly in the manner in which they generate interesting behavioural patterns. COBWEB, the conceptual clustering algorithm, generates a hierarchy, which can be used to select communities at different levels of generality. Furthermore, it generates disjoint communities, which map each user uniquely to a community. This might be desirable in services where each user community needs to be treated separately. However, it will not be desirable in many services, where users naturally belong in more than one community. In those cases the cluster mining method seems more appropriate. Another advantage of the cluster mining method is that it generates behavioural patterns directly, without the need for community characterisation.

In addition to those basic differences of the two methods, one of the goals of the three case studies was to perform a comparison of the two methods in terms of objective, measurable criteria. The criteria that we used were: the distinctiveness between the community models generated by each method and the extent to which these models cover the descriptive features of the domain, e.g. Web pages in a site or news categories in a news-filtering system. According to these criteria, the models generated by cluster mining seemed better than those generated by COBWEB. This is mainly due to the directness with which cluster mining addresses the problem of pattern discovery, in contrast to COBWEB, which requires a post-processing stage, in order to characterise the generated communities.

Despite the large differences, both in the design and in the performance of the two methods, the examination of indicative models generated in each case study, showed substantial overlap between the resulting models. This overlap can be interpreted as confirmation for the significance of the discovered patterns. The cases in which the results of the two methods do not overlap are also interesting, because they indicate that the two methods complement each other. For instance, conceptual clustering was able to isolate the users who cannot easily be assigned to a community, while with cluster mining we were able to identify patterns that cut across the disjoint communities generated by conceptual clustering. Therefore, the combination of the two methods seems to be an interesting option for further research.

Another interesting idea would be the use of machine learning to construct user stereotypes, where a stereotype is a community model, enriched with personal information about the user. The conceptual relationship between stereotypes and communities suggests that some of the work presented in this paper will also be of use there. However, personal information is sensitive and hard to acquire on Internet services.

A more substantial deviation from the work presented here, would be to extend the scope of the work to other types of on-line service. An interesting example would be to study the users of an Internet Service Provider, who form a less homogeneous group than the users of a specific Web site. Furthermore, it would be interesting to model other parameters of user behaviour than ‘interest’. E-learning applications provide an example, where community modelling could be based on the knowledge level of the users. The fact that the learning methods presented here do not depend on the semantics of the problem representation, suggest that they would be directly applicable to such a problem, given the appropriate transformation of the usage data to a training set.

Concluding, this paper suggests the use of community models, as a valuable tool in analysing the behaviour of users of Internet-based services. Furthermore, it suggests the use of learning methods to construct these models from usage data and compares the performance of two very different methods. The three case studies presented here provide a thorough account of the problems that are encountered in this process and lead to a number of interesting conclusions about the performance of the two methods on different types of service. However, the most important conclusion is that the discovery of behavioural patterns for user communities, with the use of learning methods, is feasible and can provide very valuable information to the users and the provider of an information service.

Acknowledgements

We would like to thank the service provider, who prefers to remain anonymous, for the dataset used in case study B, as well as the Computer Science Institute of FORTH for providing us with some of the data for case study A. We would also like to thank the GMD machine learning group for their public version of COBWEB, Mike Perkowitz and Pat Langley for useful comments on the work and George Petasis, Panagiotis Tzitziras and Victoria Malaveta for their assistance in the experiments. The work presented here is part of a long-term effort, which has been financed by a number of research projects, including the EC-funded project ECRAN (LE-2110), the project MITOS, funded by the Greek government and the project KOINOTITES, funded by NCSR "Demokritos".

References

Ardissono, L., Goy, A., 2000. Tailoring the Interaction with Users in Web Stores. User Modeling and User-Adapted Interaction 10 (4), 251-303.

Balabanovic, M. Shoham, Y., 1997. Content-Based, Collaborative Recommendation. Communications of the ACM 4 (3), 66-72.

Balabanovic, M., Shoham, Y., 1995. Learning Information Retrieval Agents: Experiments with Automated Web Browsing. Proceedings AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Resources.

Basu, C., Hirsh, H., Cohen, W., 1998. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. Proceedings Fifteenth National Conference in Artificial Intelligence (AAAI'98).

Benaki, E., Karkaletsis, V. Spyropoulos, C.D., 1997. Integrating User Modelling Into Information Extraction: The UMIE Prototype. Proceedings Sixth International Conference on User Modeling (UM'97), pp. 55-57.

Billsus, D., Pazzani, M., 1999. A Hybrid User Model for News Story Classsification. Proceedings Seventh International Conference on User Modelling (UM'99), pp. 99-108.

Billsus, D., Pazzani, M., 2000. User Modeling for Adaptive News Access. User Modeling and User-Adapted Interaction 10 (2-3), 147-180.

Bloedorn, E., Mani, I., MacMillan, T.R., 1996. Machine Learning of User Profiles: Representational Issues. Proceedings Thirteen National Conference on Artificial Intelligence (AAAI'96), pp. 433-438.

Brajnik, G., Tasso, C., 1994. A Shell for Developing Non-monotonic User Modelling Systems. International Journal of Human-Computer Studies 40, 31-62.

Brajnik, G., Guida, G. Tasso, C., 1987. User Modelling in Intelligent Information Retrieval. Information Processing and Management 23, 305-320.

Breese J.S., Heckerman, D., Kadie, C., 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI'98).

Bron, C., Kerbosch, J., 1973. Finding all cliques of an undirected graph. Communications of the ACM 16(9), 575-577.

Brusilovsky, P., Schwarz, E., 1997. User as Student: Towards an Adaptive Interface for Advanced Web Applications. Proceedings Sixth International Conference on User Modelling (UM'97), pp. 177-188.

Chen, L., Sycara, K, 1998. WebMate: A Personal Agent for Browsing and Searching. Proceedings Autonomous Agents'98 Conference.

Chin, D.N., 1989. KNOME: modelling what the user knows. In: A. Kobsa and W. Wahlster (Eds.). User models in dialog systems. Springer-Verlag, Berlin, pp. 74-107.

Cooley, R., 2000. Web Usage Mining: Discovery and Applications of Interesting Patterns from Web Data. Ph.D Thesis, University of Minnesota, Department of Computer Science and Engineering.

Esposito, F., Malerba, D., Semeraro, G., Fanizzi, N. and Ferilli, S., 1998. Adding Machine Learning and Knowledge Intensive Techniques to a Digital Library Service. International Journal on Digital Libraries 2, 3-19.

Fisher, D., 1987. Knowledge Acquisition via Incremental Conceptual Clustering. Machine Learning 2, 139-172.

Fisher, D., 1996. Iterative Optimization and Simplification of Hierarchical Clusterings. Journal of Artificial Intelligence Research 4, 147-179.

Fisher, D., Pazzani, M., 1991. Computational Models of Concept Learning. In Fisher, D., Pazzani, M., Langley, P. (Eds.). Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann, San Mateo, CA, pp 3-43.

Gluck, M. A., Corter, J.E, 1985. Information, Uncertainty and the Utility of Categories. Proceedings Seventh Annual Conference of the Cognitive Science Society, Lawrence Erlbaum Associates, 283-287.

Joachims, T., Freitag, D., Mitchell, T., 1997. WebWatcher: A tour guide for the World Wide Web. Proceedings Fifteenth International Joint Conference in Artificial Intelligence (IJCAI'97).

Kay, J., 1995. The um Toolkit for Cooperative User Modelling. User Modelling and User Adapted Interaction 4, 149-196.

Langley, P., 1999. User Modelling in Adaptive Interfaces. Proceedings Seventh International Conference on User Modelling (UM'99), Banff, Canada, pp.357-370.

Maes, P., 1994. Agents that Reduce Work and Information Overload. Communications of the ACM 37(7), 31-40.

Mamdani, A., Pitt, J., Stathis, K., 1999. Connected Communities from the standpoint of Multi-agent Systems. Journal of New Generation Computing, Special Issue on New Challenges in Intelligent Systems, Toyaki Nishida (Ed), 17(4):381-393.

Moukas, A., 1997. Amalthaea: Information Discovery and Filtering using a Multiagent Evolving Ecosystem. Applied Artificial Intelligence: An International Journal 11(5), 437-457.

Miller, G., 1990. Wordnet: an on-line lexical database. International Journal of Lexiocography 3(4).

Müller, M., 1999. Inducing Conceptual User Models. ABIS-99, 7. GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen.

Orwant, J., 1995. Heterogeneous Learning in the Doppelgänger User Modeling System. User Modelling and User-Adapted Interaction 4, 107-130.

Paliouras, G., Papatheodorou, C., Karkaletsis, V., Spyropoulos, C.D., Malaveta, V., 1998. Learning User Communities for Improving the Services of Information Providers. Proceedings Second European Conference on Digital Libraries. Lecture Notes in Computer Science 1513. Springer-Verlag, Berlin, pp. 367-384.

Paliouras, G., Papatheodorou, C., Karkaletsis, V., Spyropoulos, C.D., Tzitziras, P., 1999. From Web Usage Statistics to Web Usage Analysis. Proceedings IEEE International Conference on Systems Man and Cybernetics, pp. II-159-164.

Paternò, F., Manchini, C., Alkemade, F., 1999. Designing Web User Interfaces for Museum Applications to Support Different Types of Users. Proceedings Museums and the Web Conference, pp. 75-86.

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

Pennock, D., Horvitz, E., Lawrence, S., Lee Giles, C., 2000. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. Proceedings Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000), pp. 473-480.

Perkowitz, M., Etzioni, O., 1998. Adaptive Web Sites: Automatically synthesizing Web pages. Proceedings Fifteen National Conference in Artificial Intelligence (AAAI 98).

Perkowitz M., Etzioni, O., 1999. Adaptive Web Sites: Conceptual Cluster Mining. Proceedings Sixteenth International Joint Conference in Artificial Intelligence (IJCAI'99), pp. 264-269.

Raskutti, B., Beitz, A., 1996. Acquiring User Preferences for Information Filtering in Interactive Multi-Media Services. Proceedings Pacific Rim International Conference on Artificial Intelligence, pp. 47-58.

Resnick, P., Varian H.R., 1997. Recommender Systems. Communications of the ACM 4(3), 56-58.

Rich, E., 1983. Users are Individuals: Individualizing User Models. International Journal of Man-Machine Studies 18,199-214.

Schwartz, E.I., 1997. Webonomics. Browdway Books, New York.

Thompson, K., Langley, P., 1991. Concept Formation in Structured Domains. In Fisher, D., Pazzani, M., Langley, P. (Eds.). Concept Formation: Knowledge and Experience in Unsupervised Learning. Morgan Kaufmann, San Mateo, CA, pp 127-161.