Personalised Organisation of Dynamic E–Catalogs

Hye-young Paik and Boualem Benatallah

School of Computer Science and Engineering, The University of New South Wales,
Sydney, NSW 2052, Australia
{hpaik,boualem}@cse.unsw.edu.au

Abstract. Research into personalisation issues in product catalogs has mainly been focused on recommender systems and the needs for building adaptive catalogs have been largely ignored. Catalogs are designed by system designers who have a priori expectations for how catalogs will be explored by users. It is necessary to consider how users are using catalogs since they may have different expectations. $WebCatalog^{Pers}$ proposed a design and an implementation of a system through which integrated product catalogs are continuously adapted and restructured within a dynamic environment. The adaptation of integrated catalogs is based on the observation of customers' interaction patterns. In this paper, we extend the idea further by introducing the notion of liked minded people, where the same design principle of $WebCatalog^{Pers}$ is applied to a group of people who share similar interests.

1 Introduction

The proliferation of product catalogs offered via the Web has brought about high competition between the sellers, not just in terms of providing better quality products/services, but also, in terms of efforts to attract more and more "loyal" customers. There are constant demands from consumers to have services that can deliver *tailored Web experience*. The experience can be something as casual as browsing the web, or as significant as trading stocks or purchasing goods [12].

Research into personalisation issues in product catalogs has mainly been focused on recommender systems (i.e., recommending products that a customer may want to buy) [3, 11, 1]. There has not been much effort into building adaptive product catalog which is able to reorganise itself based on the customers' needs. This is important issue because product catalogs are usually structured in a category–based hierarchy and designed in a "one–view–fits–all" fashion by a system designer who has a priori expectations for how catalogs will be explored by customers. However, customers may have different intuitions and interests. If the the expectation of the system designer differs from customers', the customers may easily be lost in the network of hyper-links, or bored by taking many access hops.

In $WebCatalog^{Pers}$ [13], we have addressed this issue by introducing the notion of *communities* and *customer interaction patterns*. In $WebCatalog^{Pers}$, product catalogs from various sources are integrated and the resulting integrated catalogs are continuously adapted and restructured based on the observation of how customers are using the

catalogs. For example, in a catalog for computer parts, assume that it is repeatedly observed that many users always used product category `RAM` right after using category `CPU`. If the administrator merges the two categories and creates a new category `CPU&RAM`, users now only need to visit this new category once for information of both products.

Although the approach presented in [13] allows organisation of product catalogs to evolve over time by restructuring themselves continuously, there is still "one–view" of product catalogs for all of the customers. In this paper, we extend the idea presented in $WebCatalog^{Pers}$ by using the notion of "like minded" people. We recognise the fact that users have different background of knowledge, level of expertise, level of interests on certain information domain. Hence, from navigation behaviour of users, we can identify group of people who have similar interests. Once such groups have been identified, we can apply the same restructuring methodology proposed in [13] to each group, which will result in each group having its own view of product catalog organisation. This enable us to have a higher degree of system adaptability by specifically focusing on certain group of people, instead of the population of the user as a whole. The objective is to have multiple views of product catalog organisation for customers with different interests rather than forcing a single view to all.

The rest of the paper is organised as follows: Section 2 briefly summarises what has been proposed in $WebCatalog^{Pers}$. Section 3 introduces the concept of multiple views of product catalogs, and the next two sections discuss the techniques to identify like minded people. Finally we present some related work and conclusion.

2 Overview of $WebCatalog^{Pers}$

Before going into the details of the extended idea, we briefly present, in this section, what has been done in [13] to make the paper self–contained.

2.1 Catalog Communities and Catalog Registration

A catalog community[1] is a container of catalogs which offer products of a common domain (e.g., community of Laptops). It provides a description of desired products without referring to actual sellers (e.g., a seller of IBM Laptops). We illustrate catalog communities with *computers and related services* domain (see Fig.1).

There are two types of relationships defined between catalog communities: *SubCommunity–Of* and *PeerCommunity–Of*. SubCommunity–Of relationships represent *specialisation* between domains of two catalog communities (e.g., `Printer` is a sub–community of `Peripherals`). We assume that, each catalog community has at most one super–community. PeerCommunity–Of relationships are viewed as a referral mechanism in that when the user can not find (or is not satisfied with) information from a catalog community, s/he can refer to other communities that the catalog community consider as its peers (e.g., community `Display` is a peer community of `VideoCard`) [2]. A weight (a real value between 0 and 1) is attached to each PeerCommunity–Of relationship to represent the degree of relevancy as a peer. Note that communities can

[1] We use the terms *catalog community* and *community* interchangeably.

[2] It should be noted that, we do not assume that the opposite (i.e., `VideoCard` is a peer community of `Display`) systematically holds.

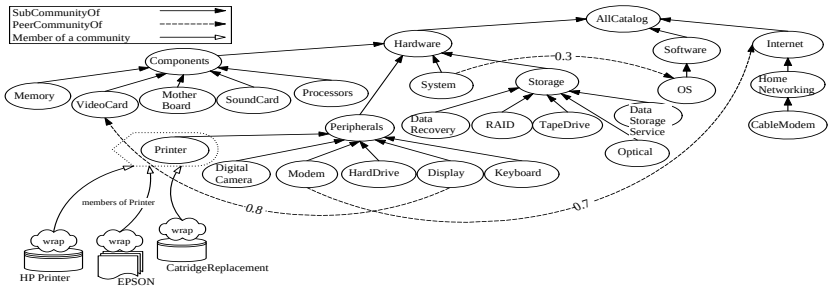

Fig. 1. eCatalogs–Net: Organising catalog communities

also forward queries to each other via PeerCommunity–Of relationship. We call this organisation of catalog communities *eCatalogs–Net*. Any catalog community that is not a sub–community of any other community is related to `AllCatalog` via SubCommunity–Of relationship. Each catalog community has a set of attributes that can be used to query the underlying catalogs. We refer to the set of attributes as *community product attributes*. For example, catalog community that represents "CDRead/Writers" would have community product attributes such as `Maker`, `ReadWriteSpeed`, `Price`, etc.

In order to be accessible through a community, product sellers need to register their catalogs with the community. A catalog provider is known to a community by providing (i) a wrapper, (ii) an exported interface, and (iii) a mapping between exported interface and community product attributes. The wrapper translates $WebCatalog^{Pers}$ queries to local queries, and output of the local queries are translated back to the format used by $WebCatalog^{Pers}$. The exported interface defines the local product attributes for querying information at the local catalog. A local catalog supplier also should provide operations, such as ordering or payment for the products. However, the focus of this paper is not on specifying transactional operations. Detailed description on provisioning such operations in the context of Web services is presented in [4]. Users may use a community to express queries that require extracting and combining product attributes from multiple underlying product catalogs (e.g., price comparison). We refer to this type of queries as *global queries*. Global querying is achieved by using community product attributes which do not directly correspond to product attributes. Therefore, when a product catalog is registered with a community, the catalog provider should also define mapping between local product attributes and community attributes. We call this mapping *Source–Community mapping*. Note that a community can be registered with another community. By doing so, the members of the first community also become members of the second community.

2.2 Permissible User Actions

Users in $WebCatalog^{Pers}$ will typically be engaged in two–step information seeking activity: (i) navigating communities for product catalogs location and semantic exploration (e.g., get communities that are relevant to selling laptops) and (ii) querying se-

lected communities or catalogs for products information (e.g., compare product prices). Users would have a specific task to achieve (e.g., product items they wish to purchase, a category of products they want to investigate) when using product catalogs. Based on this, we modelled the permissible actions for exploring eCatalogs–Net (see Table 1). By modelling user interaction actions, the system can capture them for future use.

Table 1. Permissible User Actions $\mathcal{A}$ in eCatalogs–Net

Action Name	Description
NavigateToSub(Community c)	navigate from the current community to one of its subs, `c`.
NavigateToSuper()	navigate from the current community to its super.
NavigateToPeer(Community c)	navigate from the current community to one of its peers, `c`.
LeaveCatalogCommunity()	leave the current community. The user is taken to `AllCatalog`.
ShowMembers(Constraint s)	list members of the current community matching constraint `s`.
SubmitQuery(Query q)	submits the query `q` to the current catalog community. It could be a global query using the community product attributes, or a source query concerning one member of the community.

Every time a user invokes one of the permissible actions at a catalog community, $WebCatalog^{Pers}$ keeps that event in the system log file. Each entry in the log file contains the name of an action, a user identifier (UID), a time stamp (TS), and parameters of the action. For example, the first log file entry below shows that the user, whose UID is 987, was at `Hardware` catalog community, then navigated down to its sub-community `Modem` on 15/08/2001 13:05:40 system time, etc.

(`NavigateToSub`, `UID=987`, `TS=15082001130540`, `C_FROM=Hardware`, `C_TO=Modem`)
(`NavigateToSuper`, `UID=811`, `TS=15082001130542`, `C_FROM=IBM`, `C_TO=Retailers`)

The log file is, later, organised into *sessions* and for each `SubmitQuery` action in a session, all of the product attributes selected by the query are identified. A session in $WebCatalog^{Pers}$ is an ordered sequence of actions performed by a single user, where the time difference between any two consecutive actions in the sequence should be within a time threshold, $\mathtt{T}_{\mathtt{threshold}}$ defined by an administrator.

2.3 Restructuring Operations and Predefined Interaction Sequences

The main idea is to restructure eCatalogs–Net based on the observation of user's usage behaviour. In $WebCatalog^{Pers}$, we have proposed a set of restructuring operations on eCatalogs–Net. These operations are used, for example, to change the relationships between catalog communities, remove a catalog community, or merge catalog communities. They can be performed at an administrator's own discretion, but we also proposed *predefined interaction sequences (PIS)* which provide means to observe the user's interaction patterns with eCatalogs–Net. Predefined interaction sequences represent foreseeable user's interaction behaviour, therefore can be predefined. In our approach, we use these sequences of actions to help identify situations where the organisation of an eCatalogs–Net may be improved through restructuring operations. Any particular sequence of actions with prevalent occurrences should be recognised as a recurring user

interaction pattern. Each interaction pattern identified suggests a restructuring operation. The observation of the pattern will help decide which operation to perform in order to improve the organisation of eCatalogs–Net.

For example, we have defined an operation `mergeCatComm()` which merges two communities into one (see Figure 2). An administrator will perform the operation when one of the PIS pattern

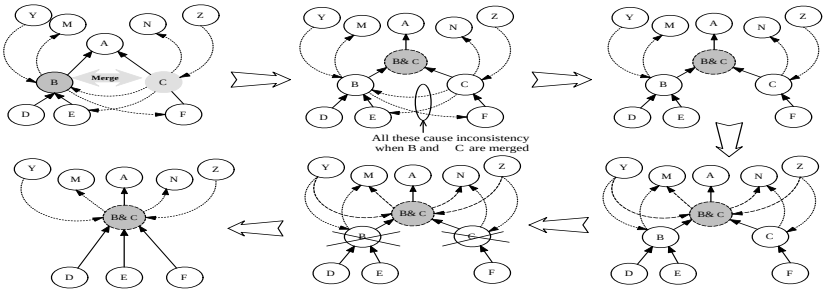

Fig. 2. Merging two catalog communities

$\texttt{PIS}_{\texttt{merge1}}$ appears frequently in the log file. $\texttt{PIS}_{\texttt{merge1}}$ identifies two sub–communities of the same super–community which are always accessed together and formally presented as follows (see Figure 3):

$$\texttt{PIS}_{\texttt{merge1}} = \langle\, \texttt{SubmitQuery}(\texttt{c}_\texttt{i}, \texttt{q}_1), \texttt{NavigateToSuper}(\texttt{c}_\texttt{i}, \texttt{c}_\texttt{k}), \texttt{NavigateToSub}(\texttt{c}_\texttt{k}, \texttt{c}_\texttt{j}), \texttt{SubmitQuery}(\texttt{c}_\texttt{j}, \texttt{q}_2) \,\rangle$$

where $\texttt{c}_\texttt{i}, \texttt{c}_\texttt{j}, \texttt{c}_\texttt{k}$ are communities in eCatalogs–Net and $(\texttt{c}_\texttt{i}, \texttt{c}_\texttt{k}), (\texttt{c}_\texttt{j}, \texttt{c}_\texttt{k})$ have the SubCommunity–Of relationship.

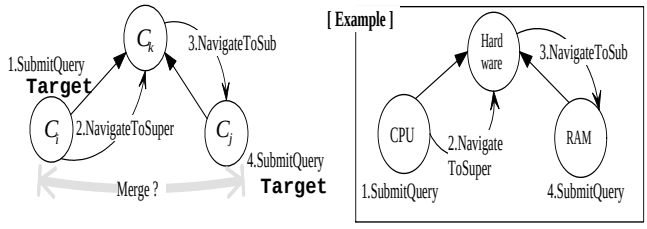

Fig. 3. PIS for Merge 1

3 Advanced Features of $WebCatalog^{Pers}$

One of the key aspects of a personalisation system is the ability to be adaptive and actively responsive to the ways the user interacts with the system. We introduced the concept of eCatalogs–Net

and how the way eCatalogs–Net is structured can be adaptively change over time based on the access pattern of users.

We used, for example, user's interaction actions traced in the log file to judge whether an existing relationship is sensible, or whether there is a need for a new relationship (or deletion of an existing relationship) between two communities. Because, if we were to decide whether community C is related to community C' or not, it makes more sense to observe what majority of users (who are actually using them) think, than to depend on subjective opinion's of an administrator.

In this section, we discuss how we can improve what we have shown so far by taking the notion of "like-minded" users into consideration. The approach in previous sections treats the whole population of users as one group and every user in the system gets the same view of the eCatalogs–Net. Whenever the system administrator updates eCatalogs–Net structure, the old view of eCatalogs–Net is disregarded and the updated version is loaded for users. However, because people have different ways of reasoning, different level of expertise or interests on a particular subject, when we observe the user's behaviour, it will be desirable to consider these differences in individual users. The differences will be reflected in users' interaction actions, hence, in the log file.

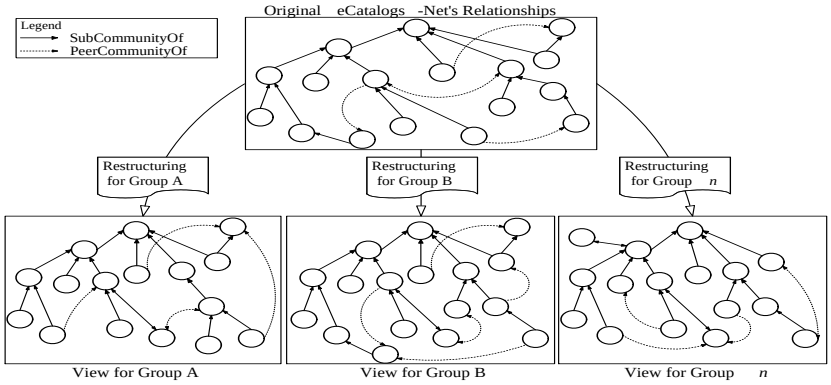

Fig. 4. Adaptive evolution of eCatalogs-net

Figure 4 illustrates the idea of adaptive evolution of eCatalogs–Net based on how a group of users perceive the structure. For example, notice that PeerCommunity-Of relationships are different from one group's view to another and also the number of communities. The new structure will reflect the user's intuition within the group. Eventually, each group would have its own view of eCatalogs–Net structure; the structure that is considered to be sensible and meaningful by the users in the group rather than by the system designer. Also, that structure itself may change over time as its population of the group changes.

Although it is likely that the user's background would reflect their interests, we can not simply rely on users' demographic data to decide like-minded groups, because it may well lead to false conclusion (e.g., not all teenagers are interested in Internet chatting). Therefore, we rely on the users' behaviour presented in the system log file as the basis for the grouping.

4 User Groups

This section of the paper is dedicated to explaining the process involved in finding the like-minded groups. Obviously, to find such groups, we first have to understand the mind of individuals. Note that we have processed the original log file into sessionised log file at this stage. The following subsections describe how we further process the sessionised log file to derive interests of individuals, and from there, identify like-minded groups among the individuals.

4.1 Building Individual User Profile

From the sessionised log file, we build individual user profiles. Building a user profile means understanding what the user's interests and preferences are when it comes to using eCatalogs–Net. It is done by finding out what the users have done in the past (i.e., from the actions they performed). This implicit gathering of user preferences allows the system to be less biased than using information gathered explicitly from user questionnaires, which tends to be incorrect, incomplete and subjective [5].

In our work, the *Individual User Profile (IUP)* contains a list of communities that the user has visited, along with an associated number indicating the level of interests for each of them[3]. We only include communities whose associated number is higher than a predefined threshold in the IUP. One can understand the individual user profile in our system as a summarisation of the user's *action trend* reflected in the sessionised log file. The actual process of deriving individual user profiles are explained in the following paragraphs.

Extracting Interest Action Set (IAS) The interests of users are measured by looking at the following actions; `ShowMembers` and `SubmitQuery`. We will call them the *Interest–Action–Set* or IAS in short[4]. Among all the permissible actions defined in the eCatalogs-Net, we consider these particular actions as an indication of user's strong interest in a community, because these actions are not normal navigation actions (hence, just passing by), rather, they are actions that require explicit and deliberate interaction with a community (e.g., submitting a query, listing a member for further investigation).

A different number value is assigned to each of these actions in IAS as a form of weight, ranging from 0 to 1. Because not all of the actions present the same level of interest of the user, we use the weight to indicate the ability of an action to reflect the user's interest. What number is assigned to which action is decided by the system designer. For example, for the two actions in the IAS used in this work, `SubmitQuery` is considered to be a stronger indication of user's interests than `ShowMembers` action, hence the `SubmitQuery` action receives a higher weight than `ShowMembers` action.

The *IAS* actions are used in a query which is applied to the sessionised log file. The queries extract all communities on which the IAS actions are performed. Using the UIDs, we would find out which communities are accessed by whom and what kind of actions are performed on them. The query that will extract all IAS actions is written as follows[5]

```
SELECT UID, Action.From AS Community, Action.Name, Count(Action.Name) AS ActCnt
FROM Sessionised_Log_File
MATCH ⟨ ShowMember—SubmitQuery ⟩
GROUP BY UID, Action.From, Action.Name
```

[3] We call the level of interests as popularity of a community to the user.

[4] More actions can be defined. Here, we only use the two.

[5] The details of query language, C-QL, is not presented in this paper.

The `MATCH`-clause matches any line in the log file that contains ShowMember or SubmitQuery action, `GROUPBY`-clause groups the output by UID, the community on which the action is performed (i.e., Action.From) and then, by action name. The system function `Count()` counts `Action.Name` which will present the number of rows (i.e., Action) in each group of tuples returned. From this query, we see that how many times each of the IAS actions has been performed on a given community by a given user. Figure 2 shows part of the result (e.g., A result on Research community by the user 987). The complete results would have outputs illustrated in Table 2 for every community that a user has visited. The result of the query is called *IAS-Query-Result* (for clarity reason, we added descriptive headings to returned tuple).

Table 2. IAS-Query-Result

UID	Community	Action	ActCnt
987	Research	ShowMembers	3
987	Research	SubmitQuery	15
987	Modem	ShowMembers	1
987	Modem	SubmitQuery	3
137	Research	SubmitQuery	30
...	...	...	.

Deriving Popularity Using the values in `ActCnt` in the IAS-Query-Result, the *popularity* is derived for each community visited by a given user. The process is to indicate the level of the user's interests on a particular community. The popularity of a given community c, by user u is denoted as $Pop_{(u,c)}$ and is calculated as follows:

$$Pop_{(u,c)} = \sum_{k=1}^{n} (Count_{a_k} \cdot W_{a_k})$$

where:

- a_k is an action in the IAS ($1 \leq k \leq n\}$).
- $Count_{a_k}$ is number of times the action (a_k) is performed on community c, given from the IAS-Query-Result (i.e., ActCnt).
- W_{a_k} is the predefined weight of the action a_k, given by an admin.

For example, let us assume that, for user u, and for the community `Research`, the counting process of IAS-Query-Result has returned what is shown in Figure 2. The popularity of community `Research` by user 987 is calculated as follows given that W_{a_k} for each action is ShowMembers = 0.5 and SubmitQuery = 1.

$$Pop_{(987,Research)} = (3 \cdot 0.5) + (15 \cdot 1) = 16.5$$

In other approaches (e.g., [17, 12]), popularity of subject items (e.g., a web page, or a product) are indicated by number of "counts" on accesses to each web page (or a category). In this paper, however, we consider not only the frequency of the subject item (i.e., community), but also the actions that are performed on it. The calculation is designed to give more weight to communities on which; (i) more significant actions, that is, actions with higher W_k, are performed, (ii) such actions are performed large number of times. The more weight a community is assigned to, the higher interests of the user is in the community.

After deriving the popularity, which calculates $Pop_{(u,c)}$ for all communities that u has visited, we can obtain an individual user profile for u.

Definition 1 (Individual User Profile). *An individual user profile (IUP) for a user u, is denoted as P_u and defined as follows:*

$$P_u = \{(c, Pop_{(u,c)}) | c \in \mathcal{C},\ 0 \leq Pop_{(u,c)} \leq Pop_{threshold}\}$$

- *c denotes a community.*
- *$\mathcal{C}$ is all communities in eCatalogs–Net.*
- *$Pop_{(u,c)}$ is the popularity of the community for user u.*
- *$Pop_{threshold}$ is the minimum popularity acceptable for a community to become part of IUP. It is given by the system designers.* □

For example, a user profile for the user with ID 987 can be;

P_{987} ={(Internet, 26), (Data_Recovery, 55), (Display, 45), (Digital_Camera, 36)
(Memory, 35), (Processors, 33), (Modem, 26)}

given that, $W_{threshold}$ = 25.

4.2 Grouping Like Minded Users

Using the individual user profiles, we can now measure similarities between users to determine the like-minded people. The idea is to partition the user profiles into clusters so that users within a cluster are close (i.e., the popularities of common communities are similar). To identify input dataset for a clustering, we arrange the individual user profiles so that they are organised as a table of tuples (like shown in Table 3). All communities that exist in the log file will be listed in the column heading. Popularity for the communities that do not appear in a user's profile are '0', i.e., for example, community `System` has never been visited by user 754. Table 3 is a sample input dataset to the clustering algorithm.

There are various clustering algorithms available, we chose K-mean [7]. We have defined our input data set for a general clustering already, hence, any algorithm can be applied. K-mean algorithm splits a set of objects into a selected number of groups. The basic idea of K-mean is to find a single partition of the data which has K number of clusters such that objects within clusters are close to each other in some sense, and those in different clusters are distant. The object of clustering, in our case, is the users and the basis of determining a cluster is the popularity of communities visited by each

Table 3. Clustering Input

community\ UID	Hardware	Software	Internet	System	Storage	RAID	Display	Printer
987	11	29	1	33	0	15	22	0
754	8	30	1	0	11	8	13	5
342	13	21	1	35	29	6	2	37
675	12	27	1	55	0	31	31	2
807	6	31	2	30	8	12	27	29
145	4	29	1	0	2	37	60	21
678	5	36	1	54	8	78	60	0
759	5	37	2	77	63	55	12	8
...	...	...	...	...	...	...	...	...

user. Hence, users (represented by UIDs) within the same cluster will be "close", in the sense that users within a same cluster would have similar popularity for communities.

From K-mean clustering, we will have K number of clusters[6]. We can presume that users belonging to the same cluster have the same kind of interests and information-seeking pattern when it comes to querying communities in eCatalogs–Net. For example, the final pass of the algorithm produces the clustering of (343), (754, 675, 987), (807, 678, 759, 145), which are UIDs, from the sample input.

4.3 Revisiting pre-process of the log file

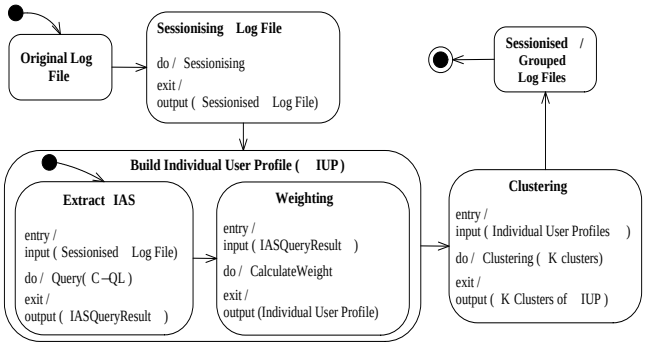

Fig. 5. Pre-process of the log file

Figure 5 summarises the pre-process of the log file; the original log file undergoes data preparation period, where it is sessionsed and global query attributes are extracted.

[6] The value of K is given by an administrator, but it also can be heuristically determined.

Further to accommodate advanced features of $WebCatalog^{Pers}$, we build user profiles based on the sessionised log file. Actions in IAS are extracted, result of which is referred to as *IAS-Query-Result* and the popularity of a community is calculated to finalise individual user profiles. Finally, through clustering of individual user profiles, the like-minded group of people are identified.

The final stage of the processed log file puts users into their similar-thinking groups, thus, it is now possible to identify each user with a group ID. Based on the group ID, we can break the sessionised log file into several sessionised log files, that is, if there is K number of groups, we can generate K number of sessionised log files, one for each group. This way, each sessionised log file concerns only the users belong to the same group. We will call these files *Sessionised Grouped Log Files (SG-Log files)*.

5 Exploring the Actions of Like-Minded Users

A SG-Log file is unique to each group, holding past actions of users belonging to the group. The same methodology presented in Sect.2.3 is applied, but this time, to the individual SG-Log files. Even though the same predefined interaction sequences are used for querying SG-Log files, the returned result from each SG-Log file will be different. The differences are caused by different group of people having different interests of subjects, and different ways of reasoning about the relationships between communities.

Let us assume that we have two different groups. To illustrate our concept, let us simply assert that one group is mainly interested in domains of 'Computer Processors', the other, in 'Computer Accessaries'. Inside eCatalogs–Net, those domains are be represented by various communities, and users from each group would search such community, and those information-seeking activities are reflected in their interactions. Hence, for a sequence like $\texttt{PIS}_{\texttt{merge1}}$ whose definition is:

$$\texttt{PIS}_{\texttt{merge1}} = \langle\, \texttt{SubmitQuery}(\texttt{c}_\texttt{i}, \texttt{q}_1), \texttt{NavigateToSuper}(\texttt{c}_\texttt{i}, \texttt{c}_\texttt{k}), \texttt{NavigateToSub}(\texttt{c}_\texttt{k}, \texttt{c}_\texttt{j}), \texttt{SubmitQuery}(\texttt{c}_\texttt{j}, \texttt{q}_2) \,\rangle$$

where $\texttt{c}_\texttt{i}, \texttt{c}_\texttt{j}, \texttt{c}_\texttt{k}$ are communities in eCatalogs–Net and $(\texttt{c}_\texttt{i}, \texttt{c}_\texttt{k}), (\texttt{c}_\texttt{j}, \texttt{c}_\texttt{k})$ have the SubCommunity–Of relationship.

When we query this sequence to the log file associated with each group, even though we used the same sequence, actual communities accessed via these actions (i.e., $\texttt{c}_\texttt{i}, \texttt{c}_\texttt{j}$) will reflect each group's main interests, that is, from one group, we would get communities that are mainly concerned with 'Computer Processors' such as community `CPU` or community `Memory`. From other group, `DigitalCamera`, `Scanner` etc. This, of course, means that communities that will be merged as a result of searching for patterns will be different from one group to the other.

The differences explained above would have significant impact on which communities the system administrator is going to change and what kind of relationships are affected for each group. This is why using separate SG-Log files, it is possible to develop a separate view of the eCatalogs–Net (Figure 4). Using the same patterns and query language as the original approach that we have discussed initially, now a different view of the eCatalogs–Net can be constructed through operating different updates

to each group, based on the pattern discovered from each *SG-Log file*. Hence, each group has a unique view of the eCatalogs–Net, which ultimately people in the group participated in creating it.

As in the original approach, the system administrator performs the pattern discovery task periodically, this means that the unique view of eCatalogs–Net for each group will change over time as well. In previous case, we assumed that new updates will replace the old view, hence there is always single view available at any time. We could make the same assumption here, that is, there is only single unique view per group at any time. However, we would like to explore the possibility of letting the system administrator decide whether to create a new version each time new updates are performed, or overwrite the existing view. Keeping the old version may be advantageous over destroying it, because the users may want to refer to the old one at some stage, or feel comfortable with the version and do not want to see any changes to it anymore.

If there are users who do not share common interests or reasoning behaviour with others, hence difficult to identify groups they belong to, we offer a default view of eCatalogs–Net. Also, it is possible to predict a situation where we have different groups, but their SG-Log files are similar, which could result in a very similar view of eCatalogs–Net. We do not consider merging the groups in such case. It is because adaptation and restructuring of eCatalogs–Net is a periodic and ongoing process. We do need to observe convincing evidence of the similarity between groups (e.g., groups demonstrating consistant similarity over a certain period of time), before they can be combined into one.

6 Related work and Conclusion

Current active approaches in personalised information delivery (in electronic forms) can be categorised in four different areas based on how the system learns individual user's preferences. They are content-based, collaborative filtering, rule-based and web usage mining. Our work is mainly related to web usage mining and collaborative filtering. We will briefly discuss the main features of each approach, then look into more specific approaches that are related to our work.

First, content-based approach has its roots in development at the mixture of information retrieval and machine learning technology [3, 11]. In this approach, the system searches for (and recommends) items similar to those the user prefers based on a comparison of content using text-learning methods. This approach, however, has difficulty capturing different types of content (e.g., images or video clips etc.) and has a problem of over-specialisation. When the system recommends items scoring highly against user's preferences, the user is restricted to seeing items similar to those already rated.

Collaborative filtering mechanism works on a foundation that, rather than finding items similar to a user has liked in the past, it searches items that other similar users have liked. Hence, this approach compute the similarity of users instead of items themselves [10, 15]. Pure collaborative filtering can solve most problems of pure content-based approach; the content is not limited to text documents because the items are recommended based on the ratings, rather than content similarity, so items could be anything (e.g., movies, books, articles etc.). Also, by selecting items liked by similar

users, the risk of being over-specialised against a single user can be reduced. However, if the number of users is small relative to the volume of items to be rated, it is possible that the coverage of ratings become very sparse and this leads to reduction of recommendable items. Another problem is that for a user whose taste is unusual compared to the rest of the user population, would not get high quality recommendation, because there are not enough similar users. Recent personalisation systems use mixture of both content-based and collaborative filtering so that they can take advantages of the approaches and overcome disadvantages of each other [3, 2].

Rule-based approach [6, 16] uses demographic or other kind of purposely collected data of users to build user profiles and then define a set of rules to tailor the content delivery based on the facts specified in the user profiles. This is direct, simple and straight forward way of personalising a web site for delivering highly relevant information. However, creation and maintenance of rules are generally manual, as the system gets complicated, there will be difficulties managing it without conflict of logics.

The effort to learn about users behavioural patterns and preferences without relying on data explicitly gathered from users (e.g., questionnaire at registration time, feedback forms) has been made in the area of web usage mining, which mines the log file produced by Web servers to discover user access patterns of Web pages. The some of the main advantages of web usage mining in personalisation area are; first, the input is not a subjective description of users (by the users themselves), thus is less likely to be biased. Secondly, since the user profiles are dynamically obtained from user's click-stream, the system performance does not suffer from "old user profile".

The application of web usage mining is diverse. The work presented in [12] uses web usage mining for recommender systems. It attempts to improve scalability of traditional collaborative filtering techniques by clustering similar user sessions based on occurrence patterns of URL references, and focus the search only in the matching clusters. Similarly, in [8], data mining association rules is used to predict presentation resource demands in interactive multimedia e-commerce catalogs. In [9], user's access histories in Web log file are clustered on a dissimilarity basis and each cluster represents a particular traversal pattern. When a user logs in, the Web server determines which cluster of traversal pattern the user belongs to and then, present links to web pages that the user is likely to visit. Also, another adaptive web server work in [14], uses data mining approach to find co-occurring pages in user visits to build index pages. But it goes further to cluster the co-occurring pages into a set of links that are conceptually related. That is, each cluster comprises index pages that both pure (containing no inappropriate links regards to the concept that the cluster is representing) and complete (containing all links that accord with the concept).

The salient differences between the above works and ours are three folds: First, we specifically model permissible actions of users to track them dynamically. Hence, the log file is not a series of web pages access (which cannot give semantic meaning behind the access to the page), rather it is a series of user actions. Using user actions carrying explicit meaning of them enables us to infer more precise semantics behind their behavioural pattern. Secondly, although it is possible to feed our log file data to data mining algorithm to discover patterns that may exists in the data, we use quite the opposite approach, in which we have pre-defined patterns that we are interested in

searching and query those to see if pre-defined patterns exist. This allows us to develop fast personalisation system which reacts to the user navigation patterns. Finally, we used the user navigation mining concept in a novel application domain, re-organising product catalogs based on the user's access patterns.

References

[1] Amazon.Com Inc. http://www.amazon.com.

[2] C. C. Aggarwal and P. S. Yu. Data Mining Techniques for Personalization. *Bulletin of the Technical Committee on Data Engineering*, 23(1), March 2000.

[3] M. Balabanovic and Y. Shoham. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, 40(3), March 1997.

[4] B. Benatallah, M. Dumas, Q. Z. Sheng, and A. H.H. Ngu. Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *Proc. of the International Conference on Data Engineering*, San Jose, USA, February 2002.

[5] K. Garvie Brown. Content Personalisation Overview, December 1999. http://www.comnet.ca/ gbrown/personalisation/persintro.htm.

[6] S. Ceri, P. Fraternali, and S. Paraboschi. Data-Driven One-to-One Web Site Generation for Data Intensive Applications. In *Proc. of the 25th VLDB Conference (VLDB'99)*, Edinburgh, Scotland, 1999.

[7] J. A Hartigan. *Clustering Algorithms*. WILEY Publication, 1975.

[8] S. Hollfelder, V. Oria, and M. Tamer zsu. Mining User Behavior for Resource Prediction in Interactive Electronic Malls. In *Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 863–866, New York, July 2000.

[9] T. Kamdar and A. Joshi. On Creating Adaptive Web Servers Using Weblog Mining. Technical Report TR-CS-00-05, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD, November 2000.

[10] J. A. Konstan, B. N. Miller, and D. Maltz. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3), March 1997.

[11] D. Mladenic. Text-Learning and Related Intelligent Agents:A Survey. *IEEE Intelligent Systems*, 14(4):44–54, July/August 1999.

[12] B. Mobasher, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. *Communications of the ACM*, 43(8), August 2000.

[13] H. Paik, B. Benatallah, and R. Hamadi. Usage–Centric Adaptation of Dynamic E–Catalogs. In *Proc. of 14th International Conference on Advanced Information Systems Engineering*, Toronto, Canada, May 2002.

[14] M. Perkowitz and O. Etzioni. Adaptive Web Sites. *Communications of the ACM*, 43(8), August 2000.

[15] J. Rucker and M. J. Polanco. Siteseer: Personalized Navigation for the Web. *Communications of the ACM*, 40(3), March 1997.

[16] BEA Systems. BEA WebLogic Commerce and Personalisation Server, 2001. http://www.bea.com/products/weblogic/commerce/index.shtml.

[17] K. Wu, C. C. Aggarwal, and P. S. Yu. Personalization with Dynamic Profiler. Technical Report IBM Research Report, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, New York, 2001.