Towards personalised web intelligence

Ah-hwee TAN
Singapore Management University, ahtan@smu.edu.sg

Hwee-Leng ONG

Hong PAN

Jamie NG

Qiu-Xiang LI

DOI 10.1007/s10115-003-0130-9
Springer-Verlag London Ltd. © 2004
Knowledge and Information Systems (2004) 6: 595–616

Knowledge and Information Systems

Towards personalised web intelligence

Ah-Hwee Tan, Hwee-Leng Ong, Hong Pan, Jamie Ng, Qiu-Xiang Li

Institute for Infocomm Research, Singapore

Abstract. The Flexible Organizer for Competitive Intelligence (FOCI) is a personalised web intelligence system that provides an integrated platform for gathering, organising, tracking, and disseminating competitive information on the web. FOCI builds personalised information portfolios through a novel method called User-Configurable Clustering, which allows a user to personalise his/her portfolios in terms of the content as well as the organisational structure. This paper outlines the key challenges we face in personalised information management and gives a detailed account of FOCI's underlying personalisation mechanism. For a quantitative evaluation of the system's performance, we propose a set of performance indices based on information entropy that measures the degree of matching between a system-generated cluster structure and a user-preferred category organisation. Experimental results of a case study show that FOCI's personalisation increases the degree of matching tremendously after a reasonable number of operations. In addition, the personalised portfolios can be used to track and organise new information with a good level of performance.

Keywords: Web intelligence; Clustering; Personalization; Information management

1. Introduction

Web intelligence can be defined as the process of scanning and tracking information on the World Wide Web so as to gain competitive advantages. In the knowledge-based era, it has become increasingly risky to do business without intelligence. With the popularisation of the web, one can obtain a tremendous amount of information from online sources readily. However, it is still very labour intensive to compile and organise such information into serviceable reports.

Popular internet search engines, such as Yahoo!, Excite, AltaVista, and Lycos, retrieve documents upon user search queries but do not organise the search results. More sophisticated tools such as Copernicus, BullsEye, and NorthernLight organise search results into automatically generated folders to facilitate navigation and browsing. However, in typical clustering systems, such as K-means (Duda and Hart 1973),

Received 6 January 2002
Revised 26 July 2002
Accepted 29 October 2002
Published online 10 May 2004

Adaptive Resonance Theory networks (Carpenter and Grossberg 1987a; Carpenter and Grossberg 1987b), and Self-Organizing Maps (Kaski et al. 1996; Kohonen 1988), users have very little control on how the information is organised, and the information clusters generated may not match the user's requirements. As a consequence, internet search tools are used mainly for gathering purposes only. Serious users, such as intelligence scouts, still have to manually compile the materials according to their needs and preferences. This can be a painstaking process, especially when the information has to be updated frequently.

To alleviate the above problem, the Flexible Organizer for Competitive Intelligence (FOCI) (Tan et al. 2001) was proposed to bridge the gap between raw search results and organised competitive information by providing an integrated platform that supports the key activities in a competitive intelligence cycle. FOCI constructs information portfolios by gathering and organising on-line information into automatically generated folders. A user, upon inspecting the information groupings, can then modify the structure according to his/her requirements and preferences through a suite of cluster manipulation functions. In FOCI, personalisation is achieved through a method known as User-Configurable Clustering (Tan and Pan 2002), which incorporates user preferences in an information-clustering system. Through an interactive process of clustering, personalisation, and discovery, a user turns an automatically generated cluster structure into his/her preferred organisation. The personalised portfolios can be constantly updated by tracking and organising new information automatically. The portfolios thus function as "living reports" that can be published and shared by other users. In all, the system provides an environment for gathering, organising, tracking, and publishing of competitive information on the web.

The rest of this article is organised as follows. Section 2 presents the FOCI architecture and a brief description of its main components. Section 3 discusses the challenges in personalised information management and motivates our choice of the information-clustering engine. Section 4 presents in detail the User-Configurable Clustering algorithm and the implementation of the various personalisation functions. Section 5 presents a case study in which we quantitatively evaluate the performance of FOCI in learning user preferences and tracking new information. Section 6 reviews related work in the area of personalisation. The final section summarises and highlights future work.

2. FOCI

Referring to Fig. 1, FOCI comprises an information-gathering module for retrieving and integrating online information from diverse sources, a content-management module for organising and personalising portfolios, a content-mining module for analysing information portfolios, a content-publishing module for sharing portfolios, and a user-interface module for graphical visualisation and user interaction. We briefly describe the key functions supporting a competitive intelligence cycle below.

1. **Information gathering:** The information-gathering module allows a user to build an information portfolio by sending search queries to major internet search engines and searchable news sites, and integrating the search results returned. The user can also insert his/her own links or documents not found in the search results into the portfolio directly. An automatic *tracking* function monitors a selected set of online sources and updates the portfolio with the new content at a user-specified interval.

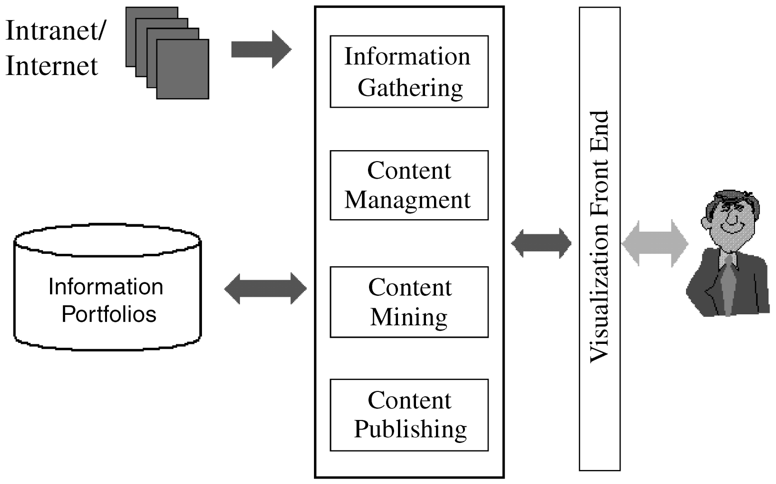
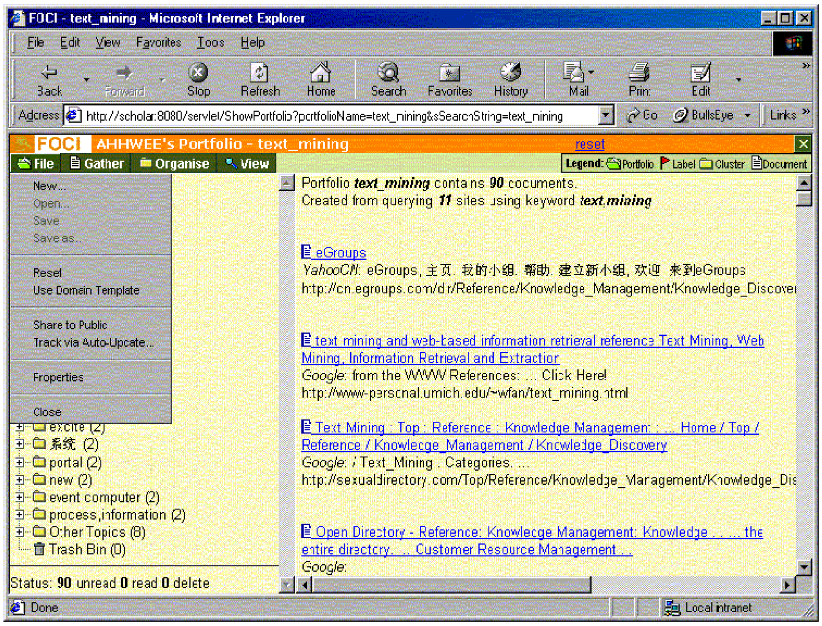


Fig. 1. The FOCI system architecture

2. **Content management:** The content-management module provides a host of utilities for a user to organise and manage his/her competitive information in the preferred manner. A domain-specific template is also available for organising information into predefined sections. In the *Technology* domain, for example, a recommended template organises competitive information into *news*, *market*, *company*, *resources*, and *events*. Coupled with an automatic clustering engine, FOCI allows a set of personalisation functions such as labelling, adding, deleting, merging, and splitting of clusters. Additional functions include annotation and deletion of documents, clusters, and portfolios.
3. **Content mining:** The content-mining module extracts key attributes from the raw information content and transforms them into an intuitive format for information discovery. The key analysis functions include topic detection/tracking, trend analysis (Kanagasa and Tan 2001), and link association. This module is currently under development and has not yet been integrated.
4. **Content publishing:** The content-publishing module handles the permission control of the individual information portfolios so that a user can elect to release his/her portfolios for public access. Various views are also supported for presenting portfolios in different levels of detail.

The FOCI server runs on a UNIX SOLARIS workstation. The system is available for public access at http://textmining.krdl.org.sg/FOCI. As the user interface is based on Servlets and dynamic HTML, the application is currently accessible through Internet Explorer (IE) version 5.5 and above only.

Figure 2 provides a screen shot of FOCI in action. The interface shows an opened portfolio consisting of clusters/folders on the left panel and a listing of search results on the right. When a cluster is selected on the left panel, the corresponding list of search results belonging to that cluster will be displayed on the right panel. There are four main menus: File, Gather, Organize, and View. The *File* menu provides options to reset, publish, print, share, or track this portfolio. The *Gather* menu provides facilities to add new search results or specific URL's to this portfolio. The *Organize* menu calls on the personalization functions to create a personalised portfolio. For convenience, a right click on the left panel of clusters will also yield this set of


FOCI - text_mining - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit
Address http://scholar:8080/servlet/ShowPortfolio?portfolioName=text_mining&sSearchString=text_mining
Go BullsEye Links
FOCI AHHWEE's Portfolio - text_mining
reset
File Gather Organise View
Legend: Portfolio Label Cluster Document
New...
Open...
Save
Save as...
Reset
Use Domain Template
Share to Public
Track via Auto-Update...
Properties
Close
excite (2)
系统 (2)
portal (2)
new (2)
event computer (2)
process,information (2)
Other Topics (8)
Trash Bin (0)
Done
Local intranet

Status: 90 unread 0 read 0 delete

Portfolio *text_mining* contains 90 documents.
Created from querying 11 sites using keyword *text mining*

eGroups
YahooCN: eGroups, 主页, 我的小组, 帮助, 建立新小组, 欢迎 来到eGroups
http://cn.egroups.com/dir/Reference/Knowledge_Management/Knowledge_Discover

text mining and web-based information retrieval reference Text Mining, Web Mining, Information Retrieval and Extraction
Google: from the WWW References: ... Click Here!
http://www-personal.umich.edu/~wfan/text_mining.html

Text Mining : Top : Reference : Knowledge Management : ... Home / Top / Reference / Knowledge_Management / Knowledge_Discovery
Google: / Text_Mining . Categories. ...
http://sexualdirectory.com/Top/Reference/Knowledge_Management/Knowledge_Dis

Open Directory - Reference: Knowledge Management: Knowledge ... ... the entire directory. ... Customer Resource Management ...
Google:

Fig. 2. A screen shot of FOCI

menu. The *View* menu provides various options to toggle among different parts and levels of the display.

Due to space constraints, the rest of this paper will focus only on FOCI's personalised content-management capabilities. Please refer to Ong et al. (2001) for a more detailed description of how a user may use the various functionalities of FOCI to gather, organise, disseminate, and track information on the web.

3. Approach and challenges

Our goal is to provide an integrated environment in which a user can transform a list of information (in this case, the search results obtained from the web) into an information portfolio organised according to the user's preferences. In contrast to typical bookmark functions of major web browsers, we use a clustering algorithm to identify the natural groupings of the documents and empower the user with cluster personalisation capabilities to transform the machine-generated groupings into his/her preferred organisation. We elaborate on the key challenges of such an approach below.

Incremental clustering: The system has to organise the documents as automatically as possible into groups based on their similarities. This helps to condense a long list of documents into a reasonable number of clusters so that it is easier for a user to browse and locate the information of interests. Another key benefit of clustering, as popularly used in the field of data mining, is to uncover interesting groupings previously unknown to a user. For the purpose of tracking, the clustering process has to be stable and incremental in the sense that clustering of new information will not erase or affect the existing groupings.

Personalization: The system needs to incorporate instructions from a user on how a portfolio is to be organised. Besides the typical functions of labelling and annotating clusters, a user may want to define his/her own groupings for organising the documents and/or modify existing clusters. Sufficient flexibility should be provided to transform a machine-generated cluster structure into a user-preferred organisation. As we need both personalisation and clustering, the system is required to perform supervised as well as unsupervised learning at the same time.

Interactivity: Our aim is to support personalised web intelligence in real time. This implies that clustering and personalisation of a portfolio must be done in an online and interactive manner. Specifically, the system should be able to incorporate a user's preference on-the-fly in real time and re-organise the cluster structure in a matter of seconds.

Based on the above requirements, we adopt a class of predictive self-organising networks, known as the Adaptive Resonance Associative Map (ARAM) (Tan 1995), as the clustering engine. ARAM extends a family of unsupervised learning systems, namely Adaptive Resonance Theory (ART) networks, to incorporate supervisory preference signals and is capable of performing incremental supervised and unsupervised learning simultaneously. It has an advantage over traditional clustering engines, such as K-means and Self-Organizing Maps, as online incremental clustering is a critical requirement for supporting the interactive personalisation and discovery process. More importantly, the predictive self-organising network architecture is compatible with symbolic rule-based knowledge representation (Tan 1997). This means that user preference in the form of class or category assignment can be incorporated relatively easily.

Our approach to personalised information management can, in principle, be applied to other clustering algorithms, such as K-means and Self-Organizing Maps (SOM). But it will definitely involve a major extension and modification of the original algorithms. There are two key obstacles to overcome. An easier one is to extend a pure clustering system to perform both supervised learning and unsupervised learning simultaneously. Specifically, the system has to incorporate the constraint that it only groups documents with the same category labels (if given) in the same cluster. The more difficult problem, however, is to enable interactive manipulation of the cluster structure. K-means and SOM are both so-called slow learning systems that iteratively refine the cluster locations according to the data distribution. It is much more difficult to incorporate user preferences, such as user-defined clusters, into such systems.

4. User-Configurable Clustering

FOCI's content-management module (Fig. 3) comprises an information-clustering engine for clustering information based on similarities, a user-interface module for displaying information groupings and obtaining user preferences, a personalisation module for defining, labelling, and modifying the cluster structure, and a knowledge base for storing the personalised cluster structure.

The personalisation module works in conjunction with the information-clustering engine to incorporate user preferences to modify the automatically generated cluster structure. By adopting a vector-space model, we assume that each document can be encoded by an information vector and that a user preference, indicating a preferred grouping of the information, can be encoded by a preference vector. Through the

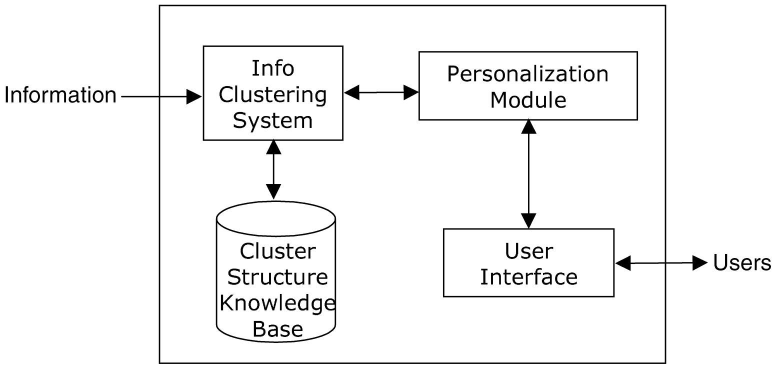


Fig. 3. FOCI's content-management module architecture

user-interface module and the personalisation module, a computer user is able to influence the organisation of the information vectors (in the form of information groupings or clusters) by indicating his/her own preferences as preference vectors. Specifically, a user can perform a wide range of cluster personalisation functions, including labelling, adding, deleting, merging, and splitting of information clusters. The customised cluster structure can be stored in the cluster structure knowledge base and retrieved at a later stage for processing new information. Based on the personalised cluster structure, new information can be organised according to the user's preferences captured over the previous sessions.

4.1. The clustering engine

An ARAM system can be visualised as two overlapping Adaptive Resonance Theory (ART) (Carpenter and Grossberg 1987a) modules consisting of two input fields F_1^a and F_1^b with an F_2 category field (Fig. 4). For User-Configurable Clustering, the F_1^a field contains the activities of the information vectors and the F_1^b field contains the activities of the preference vectors. Information clusters (represented at F_2) are created during *learning* through the synchronised clustering of the information and preference vectors. Specifically, each cluster j learns to encode the pair comprising a template information vector $\mathbf{w}_j^a$ and a template preference vector $\mathbf{w}_j^b$.

Given the information vector $\mathbf{A}$ and the preference vector $\mathbf{B}$, the system first searches for an F_2 cluster J encoding a template information vector $\mathbf{w}_J^a$ that is closest to the information vector $\mathbf{A}$ according to a choice function. It then checks whether the associated F_2 template information vector $\mathbf{w}_J^a$ and template preference vector $\mathbf{w}_J^b$ of the selected cluster match the information vector $\mathbf{A}$ and the preference vector $\mathbf{B}$ respectively, according to certain *match* criteria. If so, $\mathbf{w}_J^a$ and $\mathbf{w}_J^b$ are modified to encode $\mathbf{A}$ and $\mathbf{B}$ respectively. Otherwise, the cluster is reset and the system repeats to select another cluster until a match is found or a new cluster is created.

The ART modules used in ARAM can be ART 1 (Carpenter and Grossberg 1987a), which categorises binary patterns, or analogue ART modules such as ART 2 (Carpenter and Grossberg 1987b), ART 2-A (Carpenter et al. 1991a), and fuzzy ART (Carpenter et al. 1991b), which categorise both binary and analogue patterns. Fuzzy ARAM (Tan 1995), that is, based on fuzzy ART, is summarised below.

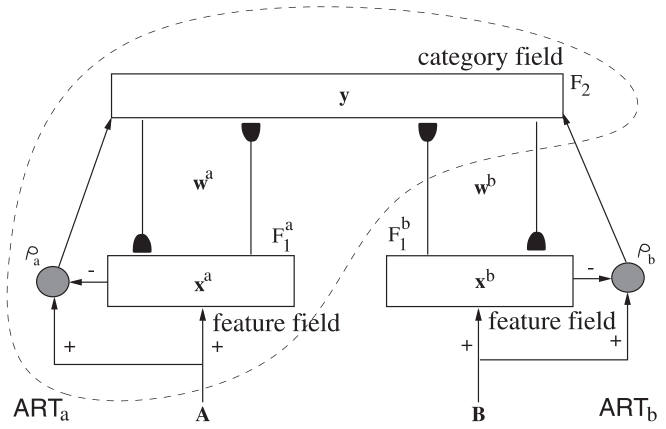


Fig. 4. The Adaptive Resonance Associative Map architecture

Input vectors: The F_1^a and F_1^b input vectors **A** and **B** represent the input information vector and the input preference vector respectively.

Template vectors: Each F_2 cluster node j is associated with two adaptive weight templates $\mathbf{w}_j^a$ and $\mathbf{w}_j^b$ for encoding the template information vector and the template preference vector respectively. Initially, ARAM contains only one uncommitted cluster node with all weights equal to one. After a cluster node is selected for encoding, it becomes *committed*.

Parameters: Fuzzy ARAM dynamics are determined by the choice parameter $\alpha_a > 0$, the learning rates $\beta_a \in [0, 1]$ and $\beta_b \in [0, 1]$, and the vigilance parameters $\rho_a \in [0, 1]$ and $\rho_b \in [0, 1]$. The choice parameter $\alpha_a > 0$ controls the bias towards choosing more specific F_2 clusters, whose template information and preference vectors have a larger norm or magnitude. The learning rates β_a and β_b control how fast the template information and preference vectors $\mathbf{w}_j^a$ and $\mathbf{w}_j^b$ adapt to the input information and preference vectors **A** and **B** respectively. The vigilance parameters ρ_a and ρ_b determine the criteria for a satisfactory match between the input and the template information and preference vectors respectively.

Category choice: Given an information vector **A** with an associated preference vector **B**, the system first searches for an F_2 cluster J encoding a template information vector $\mathbf{w}_J^a$ that is closest to the input information vector **A**. Specifically, for each F_2 cluster j, ARAM computes a choice function T_j defined by

$$T_j = \frac{|\mathbf{A} \wedge \mathbf{w}_j^a|}{\alpha_a + |\mathbf{w}_j^a|}, \tag{1}$$

where the fuzzy AND operation $\wedge$ is defined by

$$(\mathbf{p} \wedge \mathbf{q})_i \equiv min(p_i, q_i), \tag{2}$$

and where the norm $|.|$ is defined by

$$|\mathbf{p}| \equiv \sum_i p_i \tag{3}$$

for vectors **p** and **q**.

The system is said to make a choice when at most one F_2 node that has the maximal choice function value can become active. The choice is indexed at J where

$$T_J = \max\{T_j : \text{for all } F_2 \text{ node } j\}. \tag{4}$$

Resonance or reset: Upon selecting a winning cluster node J, a *template-matching* process verifies whether the template information vector $\mathbf{w}_J^a$ and the template preference vector $\mathbf{w}_J^b$ of the selected category J match well with the input information vector $\mathbf{A}$ and the input preference vector $\mathbf{B}$ respectively. Specifically, we check whether the *match functions*, m_J^a and m_J^b, meet their respective vigilance criteria as follows:

$$m_J^a = \frac{|\mathbf{A} \wedge \mathbf{w}_J^a|}{|\mathbf{A}|} \geq \rho_a \qquad \text{and} \qquad m_J^b = \frac{|\mathbf{B} \wedge \mathbf{w}_J^b|}{|\mathbf{B}|} \geq \rho_b. \tag{5}$$

Whereas the choice function computes the similarity between the input vectors and the template vectors with respect to the norm of the template vectors, the match function computes the similarity with respect to the norm of the input vectors. Together, the choice and match functions work cooperatively for fuzzy ART to achieve fast, incremental, and stable learning (Carpenter et al. 1991b).

If both the matching criteria are satisfied, ARAM enters a *resonance* state. Learning then ensues, as defined below. If any of the vigilance constraints is violated, a *reset* occurs in which the value of the choice function T_J is set to 0 for the duration of the input presentation. The search process then repeats to select another new index J until a resonance is achieved.

Learning: Once the search ends, a *template-learning* process modifies the template vectors $\mathbf{w}_J^a$ and $\mathbf{w}_J^b$ of the F_2 cluster J to encode the input information and preference vectors $\mathbf{A}$ and $\mathbf{B}$ respectively. Specifically, $\mathbf{w}_J^a$ and $\mathbf{w}_J^b$ are updated according to the equations

$$\mathbf{w}_J^{a(\text{new})} = (1 - \beta_a)\mathbf{w}_J^{a(\text{old})} + \beta_a\big(\mathbf{A} \wedge \mathbf{w}_J^{a(\text{old})}\big) \tag{6}$$

and

$$\mathbf{w}_J^{b(\text{new})} = (1 - \beta_b)\mathbf{w}_J^{b(\text{old})} + \beta_b\big(\mathbf{B} \wedge \mathbf{w}_J^{b(\text{old})}\big) \tag{7}$$

respectively. For efficient coding of noisy input sets, it is useful to set $\beta_a = \beta_b = 1$ when J is an uncommitted node, and then take $\beta_a < 1$ and $\beta_b < 1$ after the cluster node is committed. *Fast learning* corresponds to setting $\beta_a = \beta_b = 1$ for committed nodes. If an uncommitted node is selected for coding (and becomes committed), a new uncommitted node is added to the system.

4.2. Encoding information vectors

We have adopted a bag-of-words approach for representing text-based documents. To perform real-time content aggregation and clustering, we estimate the content of the pages based on the information provided on the search-result pages returned by the search engines (instead of loading the original documents). In addition to the keywords contained in the titles and descriptions of links, we also make use of the URL addresses, which provide meta-information of the web pages.

For a document d, we compute a score s_i for each keyword w_i by

$$s_i = tf(w_i) * r(w_i)(1 - r(w_i)), \tag{8}$$

where the term frequency $tf(w_i)$ is the number of times the keyword w_i appears in document d and the document ratio $r(w_i)$ is computed by

$$r(w_i) = \frac{df(w_i)}{D}, \tag{9}$$

where the document frequency $df(w_i)$ denotes the number of documents in which w_i appears and D is the number of documents in the collection. The above term-weighting scheme (8) gives advantage to keywords appearing in approximately half of the documents in the collection so as to encourage a more compact cluster structure.

The information vector of the document is then obtained by

$$\mathbf{A} = (a_1, a_2, \ldots, a_M), \tag{10}$$

where

$$a_i = \frac{s_i}{s_m}, \qquad \text{where } s_m \geq s_i \text{ for all } i, \tag{11}$$

and M is the total number of keyword features selected from a document set after filtering stop words and function words.

For user-defined terms (specified through adding clusters), the feature values are furthered enhanced by

$$a_i = \begin{cases} 1 & \text{if } a_i > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

4.3. Encoding preference vectors

User preferences, in this context, are in the form of themes or category labels assigned by a user to the individual documents or clusters. All labels specified by the users are stored in a *Label Table*. For a label l, we encode a preference vector $\mathbf{B} = (b_1, b_2, \ldots, b_N)$ such that

$$b_i = \begin{cases} 1 & \text{if } w_i = l \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where w_i is the i-th entry and N is the number of labels in the table. If a user-specified label cannot be found in the table, the label is added to the table and the dimension of the preference vectors (N) is updated accordingly.

4.4. Clustering

The algorithm for clustering using ARAM is summarised in Table 1. If a prior cluster structure exists, the system loads the ARAM network before clustering. Otherwise, a new network is created which contains only an uncommitted cluster. During clustering, for each document d, a pair of vectors $(\mathbf{A}, \mathbf{E})$ is formulated, where $\mathbf{A}$ is the information vector of d and $\mathbf{E}$ is a null vector such that $E_i = 0$ for $i = 1, \ldots, N$. The vector pair is then presented to ARAM for processing according to the algorithm described in Sect. 4.1. Clustering is completed when ARAM is stable, in the sense

Table 1. Algorithm for clustering information

```
Given a set of information items,
if a predefined cluster structure exists, load ARAM network $\mathcal{N}$;
else initialise ARAM network $\mathcal{N}$.
Loop
    For each document $d$,
    1.  Derive an information vector $\mathbf{A}$ based on $\mathcal{I}$
    2.  Derive a null preference vector $\mathbf{E}$
    3.  Present $(\mathbf{A}, \mathbf{E})$ to $\mathcal{N}$ for learning
    4.  Record index of cluster $J$ encoding $\mathcal{I}$
until $\mathcal{N}$ is stable.
```

that, for a given set of vector pairs, no new cluster is created and all the weight changes in the template vectors are below a specific threshold.

With a predefined cluster structure, fuzzy ARAM organises the information according to the cluster structure. Without a predefined network structure, ARAM reduces to a pure clustering system that self-organises the information based on the similarities among the information vectors only. The coarseness of the information groupings is controlled by the ART$_a$ vigilance parameter (ρ_a).

4.5. Personalisation

ARAM can also operate in an *insertion* mode whereby a pair of information and preference vectors can be inserted directly into an ARAM network. Whereas the *learning* mode is used for clustering and obtaining the cluster assignments of the information vectors, the *insertion* mode enables a computer user to influence the clusters created by ARAM through indicating his/her own preferences in the forms of preference vectors. During *insertion*, the vigilance parameters ρ_a and ρ_b are each set to 1 to ensure that user preferences are explicitly encoded in the cluster structure. In most cases, ARAM will recruit an uncommitted F_2 cluster node to encode the information and preference vectors inserted. In the event that the input information vector is identical to the template information vector of an existing cluster and there is a mismatch between the input preference vector and the template preference vector, we force the template preference vector to equal the input preference vector. This is appropriate for the purpose of personalisation to give priority to instructions given directly by the user. We present the algorithms for performing the various cluster personalisation functions below.

4.5.1. Labelling information clusters

Associating clusters with labels or themes allows a user to "mark" specific information groupings that are of interest to the user so that the information can be found readily in the future and new information can be organised according to such information groupings. The algorithm for labelling clusters is summarised in Table 2. To associate a cluster J with a label L, we simply insert ($\mathbf{w}_J^a$, $\mathbf{B}$) into ARAM, where $\mathbf{B}$ is a preference vector representing L. Labels reflect the user's interpretation of the groupings. They are useful landmarks to the user in navigating as well as locating old and new information.

Table 2. Algorithm for labelling clusters

```
Given an ARAM network $\mathcal{N}$, a label $L$ and the index $(J)$
of the cluster for labelling,
1.  Derive a preference vector $\mathbf{B}$ based on $L$
2.  Insert $(\mathbf{w}_J^b, \mathbf{B})$ into $\mathcal{N}$
3.  Re-cluster the information vectors using the new $\mathcal{N}$
```

Table 3. Algorithm for adding clusters

```
Given an ARAM network $\mathcal{N}$, a set of keywords $K$,
and a label $L$,
1.  Derive an information vector $\mathbf{A}$ based on $K$
2.  Derive a preference vector $\mathbf{B}$ based on $L$
3.  Insert $(\mathbf{A}, \mathbf{B})$ into $\mathcal{N}$
4.  Remove unlabelled clusters from $\mathcal{N}$
5.  Re-cluster the information vectors using the new $\mathcal{N}$
```

Table 4. Algorithm for deleting clusters

```
Given an ARAM network $\mathcal{N}$ and a cluster index $J$,
1.  Derive a preference vector $\mathbf{D}$ representing Deleted
2.  Insert $(\mathbf{w}_J^a, \mathbf{D})$ into $\mathcal{N}$
3.  Re-cluster the information vectors using the new $\mathcal{N}$
```

4.5.2. Adding information clusters

A user can define and insert his/her own clusters into an ARAM network so that the information can be organised according to such information groupings. The inserted clusters reflect the user's preferred way of grouping information and are used as the default slots for organising information.

The algorithm for adding clusters is summarised in Table 3. To insert a new cluster, a pair of information and preference vectors $(\mathbf{A}, \mathbf{B})$ is first derived based on the characterising keywords (K) of the new cluster and the cluster label L. After insertion, ARAM re-generates the cluster structures by clustering all the information vectors again. With the addition of the user-defined clusters, new clusters may be generated during the re-clustering process.

4.5.3. Deleting information clusters

Deleting clusters can be considered as a special case of labelling clusters. Basically, a user can delete a cluster by associating it with a *Deleted* label. Information in a deleted cluster can then be handled separately and hidden from the user. In addition, a *deleted* cluster serves as a filter for removing unwanted information that is similar in content in the future.

The algorithm for deleting clusters is summarised in Table 4. To delete a cluster J, a pair of template information and preference vectors $(\mathbf{w}_J^a, \mathbf{D})$ is inserted into the cluster structure, where $\mathbf{w}_J^a$ is the template information vector of the cluster and $\mathbf{D}$ is derived based on the *Deleted* label.

Table 5. Algorithm for merging clusters

```
Given an ARAM network $\mathcal{N}$, a cluster label $L$, and
cluster indices $J_1, \dots, J_n$,
1.  Derive a preference vector $\mathbf{B}$ based on $L$
2.  Insert $(\mathbf{w}^a_{J_1}, \mathbf{B}), \dots, (\mathbf{w}^a_{J_n}, \mathbf{B})$ into $\mathcal{N}$
3.  Re-cluster the information vectors using the new $\mathcal{N}$
```

Table 6. Algorithm for splitting clusters

```
Given an ARAM network $\mathcal{N}$ and a set of documents $\mathcal{I}_1, \dots, \mathcal{I}_n$
encoded in a cluster $J$ associated with the cluster labels
$L_1, \dots, L_n$ respectively,
1.  Derive the information vectors $\mathbf{A}_1, \dots, \mathbf{A}_n$ based on $\mathcal{I}_1, \dots, \mathcal{I}_n$
2.  Derive the preference vectors $\mathbf{B}_1, \dots, \mathbf{B}_n$ based on $L_1, \dots, L_n$
3.  Insert $(\mathbf{A}_1,\mathbf{B}_1), \dots, (\mathbf{A}_n, \mathbf{B}_n)$ into $\mathcal{N}$
4.  Remove the cluster $J$ from $\mathcal{N}$
5.  Re-cluster the information vectors using the new $\mathcal{N}$
```

4.5.4. Merging information clusters

Merging of clusters allows a user to combine two or more information groupings (generated by clustering) into a common theme. The algorithm for merging clusters is summarised in Table 5. To merge clusters $J_1, \dots, J_n$, the algorithm first derives a preference vector $\mathbf{B}$ encoding the user-specified label L. The vector pairs $(\mathbf{w}^a_{J_1}, \mathbf{B}), \dots, (\mathbf{w}^a_{J_n}, \mathbf{B})$ are then inserted into ARAM one at a time so that the template preference vectors of the clusters are modified to encode the common theme.

4.5.5. Splitting information clusters

Splitting of clusters allows a user to reorganise an information group that he/she deems as containing diverse content into smaller clusters of specific themes. The algorithm for splitting clusters is summarised in Table 6. To split a cluster J, a user selects a number of documents $d_1, \dots, d_n$ from the cluster as the pivots and assigns them with the labels $L_1, \dots, L_n$. The algorithm then derives information and preference vector pairs, namely $(\mathbf{A}_1, \mathbf{B}_1), \dots, (\mathbf{A}_n, \mathbf{B}_n)$, where $\mathbf{A}_1, \dots, \mathbf{A}_n$ are the information vectors of $d_1, \dots, d_n$ and $\mathbf{B}_1, \dots, \mathbf{B}_n$ are the preference vectors derived from the cluster labels $L_1, \dots, L_n$. After inserting these vector pairs into the ARAM network, each individual information vector, originally in the cluster J, will be reorganised into one of the smaller clusters depending on its similarities to $\mathbf{A}_1, \dots, \mathbf{A}_n$.

5. Performance measures

Our objective of personalisation is to enable a user to modify an automatically generated cluster structure according to his/her preferred organisation. It is thus important to have an objective measure to evaluate how well the personalisation functions have helped to transform a cluster structure into a user-preferred organisation.

FOCI performs a hybrid of clustering and categorisation. For labelled clusters, we can treat all clusters under a theme or category label as a category and evaluate the system like a categorisation or classification system. Assuming that each document in a portfolio can be categorised under a user-defined theme or category, the commonly used performance measures in the field of information retrieval, including *recall*, *precision*, and F_1 measure, can be used to evaluate how well the documents in a personalised portfolio have been classified into the predefined categories. Specifically, *recall (r)* is the percentage of documents of a given category (i.e. topic) that are classified correctly, *precision (p)* is the percentage of predicted documents of a given category that are classified correctly, and the F_1 measure is defined as

$$F_1(r, p) = \frac{2rp}{r + p}. \tag{14}$$

Likewise, the same set of performance measures can be used to evaluate the performance of a personalised portfolio in tracking and organising new documents under the predefined topics.

Although the *recall* and *precision* measures are effective in evaluating how the documents are classified into the respective categories, their applicability is limited to the labelled clusters. In other words, they are insensitive to the organisation of the documents in unlabelled clusters. To have an overall assessment of the clustering quality, Boley (1998) proposed an information entropy approach to evaluate a set of information clusters according to the original category labels of the documents. For each cluster c, a cluster entropy e_c is computed by

$$e_c = -\sum_l \frac{n_{(l,c)}}{\sum_c n_{(l,c)}} log \frac{n_{(l,c)}}{\sum_c n_{(l,c)}}, \tag{15}$$

where $n_{(l,c)}$ is the number of documents in cluster c with label l. The cluster entropy evaluates how well a cluster groups similar items together. If a cluster only contains documents with one category label, the cluster entropy is zero.

The overall cluster entropy e is then given by a weighted sum of individual cluster entropies

$$e = \frac{1}{N} \sum_c n_c e_c, \tag{16}$$

where n_c is the number of documents in cluster c and N is the total number of documents.

Although the cluster entropy reflects the quality of the clusters in terms of the class homogeneity of the documents in a cluster, it does not measure the compactness of a clustering solution in terms of the number of clusters generated. A clustering system that generates many clusters would certainly have a very low cluster entropy but is not necessarily desirable. To counter this deficiency, another entropy measure is proposed herein to evaluate how the documents of the same class are represented by the various clusters created. For each class label l, we compute a class entropy e_l by

$$e_l = -\sum_c \frac{n_{(l,c)}}{\sum_l n_{(l,c)}} log \frac{n_{(l,c)}}{\sum_l n_{(l,c)}}, \tag{17}$$

where $n_{(l,c)}$ is the number of documents in cluster c with label l. The class entropy evaluates whether the documents of the same class are represented by a minimal number of clusters. For example, if all the documents belonging to a class are contained in a cluster, the class entropy is zero.

The overall class entropy $\bar{e}$ across multiple classes is then given by a weighted sum of individual class entropies

$$\bar{e} = \frac{1}{N} \sum_{l} n_l e_l, \tag{18}$$

where n_l is the number of documents with class label l.

6. Experiments

In this section, we evaluate the performance of FOCI in personalising and tracking specific topics of interests through a typical portfolio created for the purpose of competitive intelligence. First, we examine the effectiveness of the personalisation functions in customising the machine-generated cluster structure according to a user's preferred category structure. In addition, we evaluate the performance of the personalised portfolio in tracking and organising new information. In both cases, we compare the performance of ARAM with a popularly used clustering algorithm known as K-means clustering.

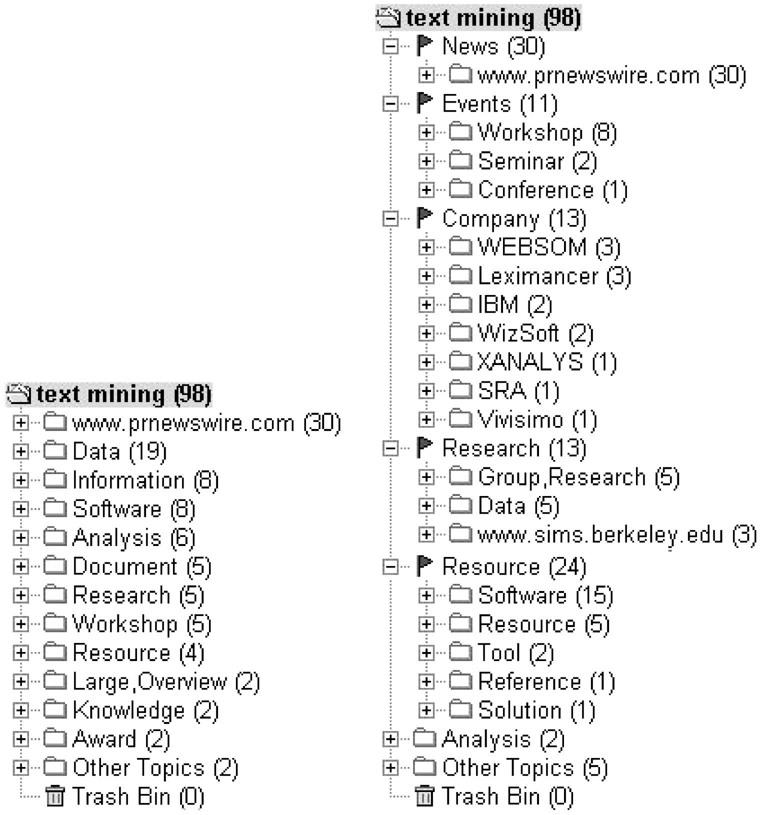
6.1. Experimental paradigm

The experiments were based on a portfolio created by a user on the topic of text mining. FOCI was first used to gather information using four search engines (Lycos, Netscape, DirectHit, and PRNewswire) with the query term of "text mining". The size of the portfolio was constrained by selecting only the top 30 hits from each search engine. After removing duplicate links, there were 98 hits. Figure 5 (left) depicts the clustering results of the documents based on a combination of URL and content-based keyword features. There were 13 clusters, each characterised by one to three keywords listed in decreasing order of importance. Two clusters, namely *www.prnewswire.com* and *data*[1], were the most prominent ones, with 30 and 19 documents respectively.

Based on the raw cluster structure generated, the subject used a combination of the various cluster manipulation functions to produce a personalized portfolio. The typical functions used were *label cluster*, *add cluster*, and *merge clusters*. As shown in Fig. 5 (right), the *www.prnewswire.com* cluster containing news articles from the PRNewswire news site was labelled under the theme of *News*. In addition, a number of user-defined cluster were created to organise the documents under the themes of *Events*, *Company*, *Research*, and *Resources*. The documents in these user-defined clusters mainly came from the original *data*, *information*, and *software* clusters in Fig. 5 (left). The user-defined clusters also helped to uncover information that was buried somewhere previously. Some of the information groupings, such as WEBSOM, were actually discovered by the system automatically during the personalisation process.

After personalisation, a new set of 63 documents was collected through three additional search engines (Businesswire, AltaVista, and MSN). Without any prior

[1] For convenience, we refer to a cluster by its first keyword.



- text mining (98)
  - www.prnewswire.com (30)
  - Data (19)
  - Information (8)
  - Software (8)
  - Analysis (6)
  - Document (5)
  - Research (5)
  - Workshop (5)
  - Resource (4)
  - Large,Overview (2)
  - Knowledge (2)
  - Award (2)
  - Other Topics (2)
  - Trash Bin (0)

- text mining (98)
  - News (30)
    - www.prnewswire.com (30)
  - Events (11)
    - Workshop (8)
    - Seminar (2)
    - Conference (1)
  - Company (13)
    - WEBSOM (3)
    - Leximancer (3)
    - IBM (2)
    - WizSoft (2)
    - XANALYS (1)
    - SRA (1)
    - Vivisimo (1)
  - Research (13)
    - Group,Research (5)
    - Data (5)
    - www.sims.berkeley.edu (3)
  - Resource (24)
    - Software (15)
    - Resource (5)
    - Tool (2)
    - Reference (1)
    - Solution (1)
  - Analysis (2)
  - Other Topics (5)
  - Trash Bin (0)

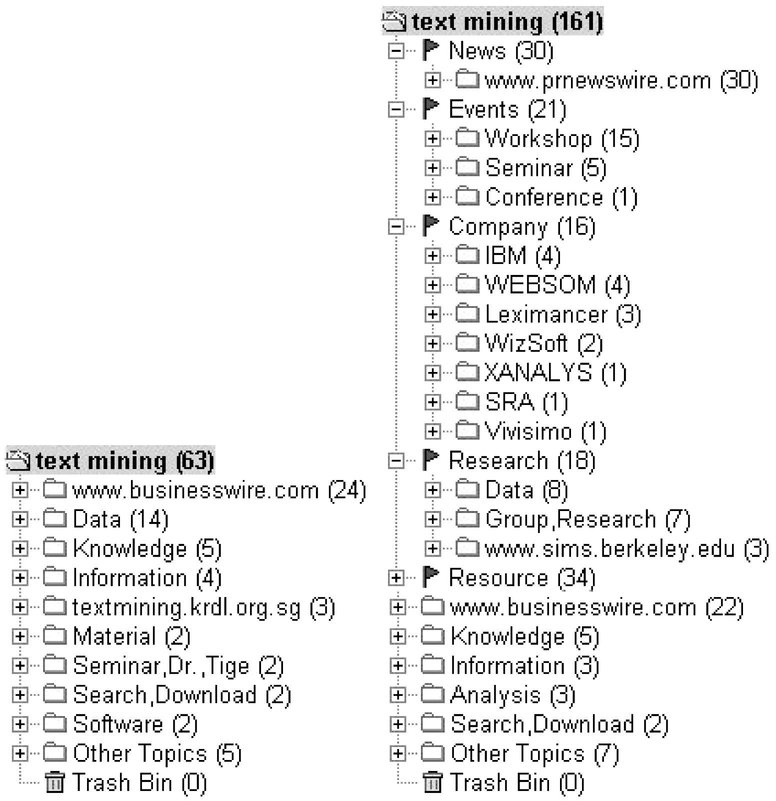
Fig. 5. Cluster structure of the portfolio before (left) and after (right) personalisation

structure, the documents would have been organised into the clusters shown in Fig. 6 (left). In contrast, Fig. 6 (right) shows the clustering result when the new documents were organised into the personalised cluster structure containing all 161 documents. A significant portion of the new information was organised under the user-defined themes. Some of the remaining clusters highlight information that did not fit into the personalised portfolio. The most prominent group was the *businesswire* cluster that contained news articles from the BusinessWire news site.

6.2. Results and discussion

For evaluation, we requested that the user provide a category label for each of the 161 documents in the portfolio. These category assignments served as the underlying targets which we used to compute the various performance indices.

Table 7 summarises the performance of K-means and ARAM on the first set of 98 documents in terms of cluster entropy and class entropy. K-means with $K = 5$ provided a balance between the cluster entropy and class entropy. However, its cluster entropy was much higher than that of ARAM. As K increased, the cluster entropy decreased but the class entropy became larger. When K was increased to 14 (the same number of clusters generated by ARAM), K-means' cluster and class entropies



```
text mining (63)
  www.businesswire.com (24)
  Data (14)
  Knowledge (5)
  Information (4)
  textmining.krdl.org.sg (3)
  Material (2)
  Seminar,Dr.,Tige (2)
  Search,Download (2)
  Software (2)
  Other Topics (5)
  Trash Bin (0)
```

```
text mining (161)
  News (30)
    www.prnewswire.com (30)
  Events (21)
    Workshop (15)
    Seminar (5)
    Conference (1)
  Company (16)
    IBM (4)
    WEBSOM (4)
    Leximancer (3)
    WizSoft (2)
    XANALYS (1)
    SRA (1)
    Vivisimo (1)
  Research (18)
    Data (8)
    Group,Research (7)
    www.sims.berkeley.edu (3)
  Resource (34)
  www.businesswire.com (22)
  Knowledge (5)
  Information (3)
  Analysis (3)
  Search,Download (2)
  Other Topics (7)
  Trash Bin (0)
```

Fig. 6. (Left) Organization of the new documents without the personalised portfolio. (Right) Organization of the new documents into the personalised portfolio

were both slightly above those of ARAM. This indicates that as a pure clustering system, ARAM's performance is at least as competitive as K-means. In terms of efficiency, both K-means and ARAM converged within a few seconds, given the small size of the portfolio. With personalisation, very significant improvement can be obtained for ARAM. The cluster entropy was practically zero, which implied that no cluster contained documents with different category labels. The class entropy was also significantly reduced from around 0.4 to slightly above 0.1. This indicates that the personalisation functions were effective in modifying the cluster structure to fit the user's preferred structure.

Table 8 shows the performance of the personalised portfolio in organising the 98 documents into the five themes in terms of recall, precision, and the F_1 measure. After personalisation, all categories achieved 100% precision, which is consistent with the zero cluster entropy. The *News* and *Events* categories also achieved 100% in recall. Only four documents were left unclassified from the *Company*, *Research*, and *Resources* categories. Note that each personalisation function results in one or more labelled clusters. This implies that the user took no more than 19 personalisation operations to organise almost all 98 documents into the five themes. In comparison, a conventional bookmark facility will require the user to perform the "file bookmark" operation about 98 times to achieve the same result.

Table 7. Performance of K-means and ARAM in terms of cluster entropy e and class entropy $\bar{e}$ on the training set. Note that as personalisation organised clusters into themes, the number of clusters/groupings actually decreased from 14 to 7 after personalization

Method	Number of clusters	Entropy e	$\bar{e}$
K-means ($K = 5$)	5	0.3545	0.3451
K-means ($K = 10$)	10	0.2348	0.4313
K-means ($K = 14$)	14	0.2058	0.4600
ARAM	14	0.1877	0.4555
ARAM (w/personalization)	7	0.0000	0.1113

Table 8. Performance of the personalised portfolio in classifying the first set of 98 documents in terms of recall, precision, and the F_1 measure

Category	Number of documents	*Recall*	*Precision*	F_1
News	30	1.00	1.00	1.00
Events	11	1.00	1.00	1.00
Company	17	0.76	1.00	0.87
Research	15	0.87	1.00	0.93
Resources	25	0.96	1.00	0.98

Table 9. Performance of K-means and ARAM on the test set in terms of cluster entropy e and class entropy $\bar{e}$. Note that ARAM was personalised on the training set only

Method	Number of clusters	entropy e	$\bar{e}$
K-means ($K = 5$)	5	0.3099	0.2603
K-means ($K = 10$)	10	0.1816	0.3880
ARAM	10	0.1920	0.4285
ARAM (w/personalisation)	11	0.1346	0.3290

Table 9 summarises the performance of K-means and ARAM on the 63 new documents in terms of cluster entropy and category entropy. With $K = 10$, the performance of K-means was roughly comparable to that of ARAM without personalisation. With personalisation, the cluster entropy and the class entropy of ARAM for the new documents were both significantly lower than those of K-means. This indicates that the personalised portfolio was able to organise the new documents in a way that was closer to the user's expectation.

Table 10 shows how well the personalised ARAM organised the 63 new documents into the five themes in terms of recall, precision, and the F_1 measure. The *Events* and *Company* categories achieved 100% in both recall and precision. This represents the best-case scenario, with the presence of "good" cluster keywords, such as "workshop" and "seminar" for *Events* and specific company names for *Company*. As the *www.prnewswire.com* cluster only tracked documents from PRNewswire, all 25 documents from another news source *Businesswire* were not classified into the

Table 10. Performance of the personalised portfolio in organising the 63 new documents in terms of recall, precision, and the F_1 measure

Category	Number of documents	*Recall*	*Precision*	F_1
News	25	0.00	–	–
Events	10	1.00	1.00	1.00
Company	3	1.00	1.00	1.00
Research	13	0.38	0.80	0.52
Resources	12	0.83	0.60	0.70

News category. This represents the worst-case scenario, in which an entire group of documents is misclassified. Nevertheless, these documents were grouped neatly into a cluster and could be organised under the *News* category by using just one *label cluster* operation. *Research* and *Resources* represent average-case scenarios, in which the categories manage to track some but not all of the relevant documents. This is perhaps due to the relatively vague nature of the categories. Note that the tracking performance of a category depends largely on the quality of its cluster keywords. The performance is expected to improve with more personalisation, which provides a form of relevance feedback.

7. Related work

Personalisation, according to Riecken (2000), is about providing a better service by understanding the needs of each individual and helping to satisfy a goal that efficiently and knowledgeably addresses each individual's needs in a given context. Personalisation has been widely used in customer relationship management. Most e-commerce companies, such as Amazon.com and eBay, maintain profiles of customers to provide personalised services and product recommendations.

In the context of information management, personalisation is usually used in three key areas: namely *filtering*, *organising*, and *display*. Personalised filtering helps to identify relevant information with respect to a user's profile and interests. The simplest form of personalised filtering systems allows a user to select from a number of predefined topics or modules. Examples of such systems include FishWrap (Chesnais et al. 1995), a personalized news system developed at MIT, and My YAHOO! (Manber et al. 2000), the personalised version of Yahoo!. Most such systems also allow a user to personalise the layout, i.e., the look and feel of his/her own pages, as well as other options, such as the schedule for updating the content. More sophisticated forms of personalised filtering systems, such as Krakatoa (Kamba et al. 1995) and SmartPush (Kurki et al. 1999), learn a user profile for each user through relevance feedback and deliver relevant documents with respect to the user profiles. FOCI belongs to this class of personalised filtering systems in the sense that the personalised information portfolios serve as user profiles for tracking new information relevant to the user's interests. Another form of personalised filtering is personalised recommendation for web navigation. A recommendation system usually suggests relevant pages by matching a user profile with that of other users in the same community. SiteSeer (Rucker and Polanco 1997) is an example of a collaborative web page recommendation system. Personal WebWatcher (Mladenic 1996) observes users of the web and recommends links in which they might be interested.

Besides filtering and recommendation, another key application area of personalisation is *organising*. Many popular internet browsers allow a user to organise bookmarks in a personalised manner. PowerBookmarks (Li et al. 1999) presents the user with advanced query, classification, and navigation features to collect bookmarks. The PAINT system (Oostendorp et al. 1994) views the internet as a file system and helps in personalising views on it. The BASAR system (Thomas and Fischer 1997) helps users to manage their personal information spaces by managing and updating links under bookmark headers. These systems are similar to FOCI in the way they help a user to collect and organise information on the web. However, in these systems, the topic trees or folders have to be predefined by the users. The systems themselves are not capable of recommending new groupings for the incoming documents even if their content deviates significantly from the predefined groupings. Compared with such systems, FOCI provides a more thorough solution that enables a user to organise information according to his/her preferred categories and at the same time, automatically discovers groupings that are novel to the user.

From the post-search clustering perspective, FOCI is similar to NorthernLight and Vivisimo, both of which are internet-based search engines that organise search results into clusters automatically. However, in these systems, the folders are generated automatically and a user is not allowed to personalise or modify the folders. By adding personalisation to clustering, FOCI provides personalised content management that simultaneously performs *classification* based on the user-defined specifications as well as *clustering* based on the natural groupings of the information.

8. Conclusions and future work

This paper has reported our recent development of a web-based intelligence system known as FOCI that enables a user to create and manage personalised information portfolios. Experimental results based on quantitative measures indicate that the personalisation functions are highly effective in modifying an automatically generated cluster structure towards a user-preferred organisation. In addition, the personalised portfolios can be used to track and organise new documents with a reasonable level of accuracy.

As we begin to release FOCI to more users for evaluation, a number of its limitations become apparent. First of all, FOCI organises a document into one and only one cluster. This is deemed a limitation for some users, who want to refer to the same document in multiple places of the portfolios. Secondly, FOCI only supports a flat level of clustering, while a user, dealing with a large number of documents, may prefer to have a hierarchical cluster structure. In addition, the system has so far adopted a simple bag-of-words approach to representing documents. Although we had attempted to use terms (containing one or more words) as the representation unit, our experiments produced many small clusters, as the occurrence frequencies of terms were significantly lower than those of words. These limitations of FOCI have raised many new challenges and will form part of our future work.

Acknowledgements. We would like to acknowledge the anonymous reviewers for their very useful comments and suggestions.

References

Boley DL (1998) Principal direction divisive partitioning. Data Min Knowl Discovery 2(4):325–344

Carpenter GA, Grossberg S (1987a) A massively parallel architecture for a self-organizing neural pattern recognition machine. Comput Vision Graphics Image Process 37:54–115

Carpenter GA, Grossberg S (1987b) ART 2: Self-organization of stable category recognition codes for analog input patterns. Appl Opt 26:4919–4930

Carpenter GA, Grossberg S, Rosen DB (1991a) ART 2-A: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks 4:493–504

Carpenter GA, Grossberg S, Rosen DB (1991b) Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Networks 4:759–771

Chesnais P, Mucklo M, Sheena J (1995) The Fishwrap personalized news system. Proceedings of the IEEE Second International Workshop on Community Networking Integrating Multimedia Services to the Home, pp 275–282, http://fishwrap.mit.edu

Duda R, Hart P (1973) Pattern Classification and Scene Analysis. John Wiley & Sons, New York

Kamba T, Bharat K, Albers M (1995) The Krakatoa Chronicle – an interactive, personalized, newspaper on the web. Proceedings of the Fourth International World Wide Web Conference, pp 159–170

Kanagasa R, Tan A-H (2001) Topic detection, tracking and trend analysis using self-organizing neural networks. Proceedings of the Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 102–107

Kaski S, Honkela T, Lagus K, Kohonen T (1996) Creating an order in digital libraries with self-organizing maps. Proceedings of the World Congress on Neural Networks, pp 814–817

Kohonen T (1988) Self-organization and Associative Memory. Springer-Verlag, Berlin

Kurki T, Jokela S, Sulonen R, Turpeinen M (1999) Agents in delivering personalized content based on semantic metadata. Proceedings of the 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace, pp 84–93

Li W-S, Vu Q, Agrawal D, Hara Y, Takano H (1999) PowerBookmarks: A system for personalizable web information organization, sharing, and management. Comput Networks 31(11–16):1375–1389

Manber U, Patel A, Robison J (2000) Experience with personalization on Yahoo! Commun ACM 43(8):35–39

Mladenic D (1996) Personal webwatcher: design and implementation. Technical report, J Stefan Institute, Department of Intelligent Systems, Ljubljana

Ong H-L, Tan A-H, Ng J, Pan H, Li Q-X (2001) FOCI: Flexible organizer for competitive intelligence. Proceedings of the Tenth International Conference on Information and Knowledge Management, pp 523–525

Oostendorp KA, Punch WF, Wiggins RW (1994) A tool for individualizing the web. Proceedings of the Second International World Wide Web Conference, pp 49–59

Riecken D (2000) Representation is everything. Commun ACM 43(8):89–91

Rucker J, Polanco MJ (1997) Siteseer: Personalized navigation for the web. Commun ACM 40(3):73–75

Tan A-H (1995) Adaptive Resonance Associative Map. Neural Networks 8(3):437–446

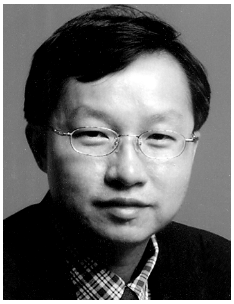
Tan A-H (1997) Cascade ARTMAP: Integrating neural computation and symbolic knowledge processing. IEEE Trans Neural Networks 8(2):237–250

Tan A-H, Ong H-L, Pan H, Ng J, Li Q-X (2001) FOCI: A personalized web intelligence system. Proceedings of the IJCAI Workhop on Intelligence Techniques for Web Personalization, pp 14–19

Tan A-H, Pan H (2002) Adding personality to information clustering. Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp 251–256

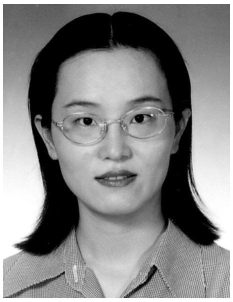
Thomas C, Fischer G (1997) Using agents to personalize the web. Proceedings of the ACM International Conference on Intelligent User Interfaces, pp 53–60

Author biographies



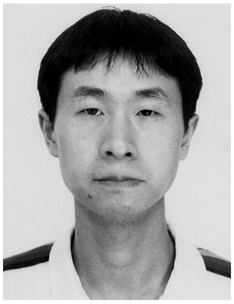
Ah-Hwee Tan is a research manager and senior member of research staff at the Institute for Infocomm Research (I2R), Singapore, where he leads R&D projects in intelligent agents, knowledge discovery, document analysis, and information mining. He received his Ph.D. in Cognitive and Neural Systems from Boston University in 1994. Prior to that, he obtained his Bachelor of Science (First Class Honours) (1989) and Master of Science (1991) in Computer and Information Science from the National University of Singapore. He has served in the program committees of numerous international conferences/ workshops and co-chaired a series of workshops on the topics of text and web mining. He is a member of the ACM and the Singapore Computer Society (SCS).

Hwee-Leng Ong is a member of research staff at I2R. She received her Bachelor of Science (Computer Science) in 1989 at the University of Southern California. Her current research interest is in information visualisation and text mining. She has led several research projects related to information visualisation, text mining, and data mining as well as industrial projects leading to the deployment or product development of the research. These include a data-mining visualisation tool as well as an internet-based expert system. She has also served on the organising committees of PAKDD97 and PRICAI98. She is a member of ACM SIGKDD and the Singapore Computer Society (SCS).



Hong Pan is a senior research engineer at I2R. She received her Bachelor of Engineering and Master of Engineering from Anhui University, China, and a Master of Engineering from Nanyang Technological University, Singapore, all in Electrical and Electronic Engineering. She has worked on text mining, knowledge discovery, and knowledge management in her current position for more than three years. Prior to this, she worked as a lecturer at the University of Science and Technology, China, for four years. Her research interests include text/image clustering and pattern recognition.

Jamie Ng is a senior research engineer at I2R. She received her Bachelor in Mechanical Engineering with Honours from Nayang Technological University in Singapore, with a special focus on human factors in design. Her current research is on user interactions with personalisable text-clustering systems. Prior to this, she worked on various human–computer interaction consultancy projects in the legal, property, education, and biotech sectors. Her research interests include human–computer interactions, text information management, and information visualisation.



Qiu-Xiang Li is a research engineer at I2R. He received his B.E. (Computer Science) from Xi'an Jiaotong University, China. His research interests include information extraction and data mining. Prior to joining LIT, he was a software engineer at Xiamen International bank, China, and was involved in server and database software development. He is a member of the Singapore Computer Society (SCS).

Correspondence and offprint requests to: Ah-Hwee Tan, Nanyang Technological University, N4-02a-32, Nanyang Avenue, Singapore 639798. Email: asahtan@ntu.edu.sg