SWAMI: A Multiagent, Active Representation of a User's Browsing Interests

Mark Kilfoil, University of New Brunswick, Canada
Ali Ghorbani, University of New Brunswick, Canada

ABSTRACT

The rapid growth of the World Wide Web has complicated the process of Web browsing by providing an overwhelming wealth of choices for the end user. To alleviate this burden, intelligent tools can do much of the drudgery-work of looking ahead, searching and performing a preliminary evaluation of the end pages on the user's behalf, anticipating the user's needs and providing the user with more information with which to make fewer, more informed decisions. However, to accomplish this task, the tools need some form of representation of the interests of the user. This article describes the SWAMI system: SWAMI stands for Searching the Web with Agents having Mobility and Intelligence. SWAMI is a prototype that uses a multi-agent system to represent the interests of a user dynamically, and take advantage of the active nature of agents to provide a platform for look-ahead evaluation, page searching, and link swapping. The collection of agents is organized hierarchically according to the apparent interests of the user, which are discovered on-the-fly through multi-stage clustering. Results from initial testing show that such a system is able to follow the multiple changing interests of a user accurately, and that it is capable of acting fruitfully on these interests to provide a user with useful navigational suggestions.

Keywords: Agent Technology, Agents, Data Mining, Information Filtering, Internet Technologies, User Information Satisfaction

INTRODUCTION

It was once possible, it is said, for one person to have read all the books that were in existence. Soon, however, that became impossible, but it was possible to index and house all the books organized by a generally acceptable set of topics. That too, is becoming rapidly unmanageable, so people turn to the opinions of people, organizations, and tools that they trust in order to determine which books they should read.

The Web has undergone a similar growth pattern, at a far greater pace. Today, we are presented with a cornucopia of information, but it has become increasingly hard to navigate and find information related to our interests. It has become necessary to find people, organizations and tools that will assist us in navigation because we can no longer read it all ourselves.

Tools have been developed to assist us. Search engines offer one-step navigation aids
in response to particular immediate information needs, but provide little in the way of ongoing assistance. Personalized, adaptive Web systems reorganize information contained within a single site to achieve local assistance, but do not offer a user any cross-site benefits.

Server-side approaches such as these only have a limited shared experience with a user, and thus they have only a narrow amount of advice they can offer. Client-side approaches are closer to the concept of a “personal assistant,” someone who knows your information history, patterns and needs and can provide ongoing assistance. However, they also observe the Web from a limited perspective.

The SWAMI system was devised to bridge the gap between client-side and service-side solutions and create an intelligent, natural, personalized assistant for user browsing. It uses the user’s own behaviour as an input to an intelligent, innovative multi-agent system which takes the drudgery out of Web navigation by reading ahead of the user, leveraging existing tools, interacting with intelligent Web sites, and sharing browsing suggestions socially.

In this article, we introduce the SWAMI system. First, we describe the domain of the problem in more detail and other approaches with their limitations. Next, we outline the high-level architecture of the full SWAMI system and how it functions. The results gathered from the currently implemented subset of the SWAMI system are discussed next. Finally, we present a summary of the benefits and drawbacks of the SWAMI approach and discuss future directions for research.

RELATED WORK

The Web is a relatively new phenomenon, and has elevated certain problems to a critical level. In this section, the two most prominent problems of Web navigation and Web personalization are discussed, and a short summary of current solutions is presented.

Web Navigation

Because of its large size, dynamic nature and inconsistent structure, the Web is difficult to navigate. “Traditional,” direct navigation approaches depend on an evaluation of the relevance of the currently viewed page as the best indicator of the value of pages pointed to by the current page. This approach relies upon the benevolence of the creator of the link (Kleinberg, 1999), and the hope that by following a series of related links the user will end up at another cluster of useful pages. This “hope” is described as the “small world” phenomenon, which suggests that a highly complex but interacting system will, over time, evolve paths of a limited number of hops between any two related pages.

When the user has discovered a page of lessening interest to him than a previous page, they return backward to an appropriately interesting (although already viewed) page and go forward from a link on that page (if there is one) until all links from that page have been exhausted, retreating back up another level. This navigation strategy is implicitly promoted by the linear nature of Web navigation tools, such as the “back” button of a Web browser.

The traditional strategy closely resembles a depth-first graph search, where leaf nodes are represented by pages of less interest. Effectively, however, the user must go “one page too far” in such a scheme, and travel deeper and deeper distances from the original page they were browsing into possibly uninteresting areas. Due to the highly connective nature of the Web, this suggests that the user will spend more time in distant pages than in pages more closely connected to the original. This, intuitively, is the opposite of the desired result, as pages directly connected to the current page are most likely to be the most relevant pages to it (Lieberman, 1995).

Search Engines

An alternative approach is to use a search engine, which, in effect, reconstructs the connection
graph of the Web, reconnecting all the distant pages together into a single layer. In this way, more relevant pages become more likely at an earlier stage of browsing. In the case of Yahoo (20042004Yahoo!, 2004!!), this rearrangement is done explicitly through a hierarchical, soft categorization of Web site links. By contrast, Google builds a response page (effectively the top-level of a tree or entrance to a graph) dynamically around a set of initial keywords in a query.

Once the user has selected a link from a search engine, however, they are out of the arena of that technology, and browsing returns to a traditional strategy. Thus, this technology produces only a one-shot or one-level navigational benefit, not ongoing navigational support.

In addition, the criteria for the evaluation of results are very specific: the keywords of the request are the only measure of relevance to the user that the system can use, although there are additional measures of the relative importance of a page (some partially dependent of the particular request made) [[REMOVED REF FIELD]]. In other words, the result evaluation does not take into account the full nature of the user, such as prior information or additional interests.

**Personalized, Adaptive Web sites**

Adaptive Web sites take a highly personalized approach. They use knowledge about the specific user to modify both the presentation of individual pages (Kobsa, Koenemann, & Pohol, 2001) and/or the navigation from one page to another (Brusilovsky, 1996). In this way, they can be seen to either add additional links between pages of relevance to the user or do a similar rearranging of the graph to the search engine, although beyond just a single level of rearrangement and navigational support. These links may come from a mining of a large set of pages within a scope (Kleinberg, 1999), from an online search and evaluation scheme (Lieberman, 1995) from an external knowledge of the structure of the domain (De Bra & Ruitter, 2001; Freitag, Joachims, & Mitchell, 1995), from other, similar users through collaborative recommendation (Cosley, Lawrence, & Pennock, 2002; Lieberman, Van Dyke, & Vivacqua, 1999; Mobasher, Dai, & Tao, 2002) or through some combination of these techniques (Balabanovic & Shoham, 1997).

Prominent examples of adaptive Web systems include WebWatcher (Joachims, Freitag, & Mitchell, 1997), AHAM (Lieberman, 1995), HiWeb (Asnicar & Tasso, 1997) and AVANTI (Fink, Kobsa, & Nill, 1997). Each of these systems provides server-side adaptive navigation or presentation based on perceived user characteristics.

Server-side solutions for adaptive Web sites also offer the possibility of collaborative recommendations, where knowledge about groups of users can be used to make suggestions to individuals who are members of a group. Groups might be arbitrarily chosen—such as the group of a person and his friends—or created through observations of common patterns of behaviour or common attributes.

Server-side solutions, however, are generally limited to a single Web site or set of close Web sites, something to which the search engine approach is not limited. Privacy concerns, however, keep users from wanting information to be shared between Web sites, particularly when personal information is being collected.

**User Modelling**

A user model is a representation of the user’s information needs. It is used to search for new information and evaluate it. Information needs can be broadly classified into three types: short-term, long-term and periodic. Short-term needs are specific and require immediate and direct response, but their significance sharply descends in time and can be quickly forgotten. Long-term needs are more general, but always have at least some value and should rarely be forgotten. Periodic needs are a compromise between both of these, having periods of high intensity interspersed with gaps of very low intensity, but should not be forgotten often.
There are two methods of discovering a user’s desired information needs: explicitly asking them to describe what they need and implicitly learning what those needs are based on behaviour. The former approach is favoured by solutions like search engines with only short-term needs in mind, and the latter approach is preferred in most other cases which use long-term needs.

Needs are not static—they change over time. Thus, a model of a user should also change to meet the current needs. Godoy and Amandi (2002) present a general architecture for discovering and maintaining a user profile in agent terms. This architecture suggests that users have multiple interests of varying levels of detail, and organizes these topics in terms of a hierarchy. It is also recognized in this architecture that user interests are not static, but tend to both change and recur over time. In their architecture, they suggest an explicit “temporal context” might be used to modify the strength of suggestions about particular topics at a given time.

Continual learning techniques are necessary for ongoing navigational support to keep up-to-date when implicit user model building is used. This approach has proven to be effective in many cases (Chan, 1999; Pazzani & Billsus, 1997; Schwab, Pohl, & Koychev, 2000).

**ARCHITECTURE**

SWAMI is a full system for providing ongoing navigational support to an end-user. The architecture covers everything from a front-end client to components responsible for representing the user’s needs to components for multiple ways to interact with sites, other tools, and other users.

This section describes the high-level architecture of SWAMI, including several of the design considerations which distinguish this work from others.

**Design Considerations**

The design process of the SWAMI system was guided by several considerations. These include:

1. **The system should not require the user to explicitly state their needs:** The system must be capable of learning the user’s needs from observation.

2. **A user may have multiple different needs:** Many systems consider the user needs to be all related to some degree. In SWAMI, it was felt that this was an unrealistic assumption, so a model that allows for multiple competing needs was developed.

3. **A user may have several related needs (“sub-interests”) within the context of a general need:** Topic hierarchies and ontologies have been proposed and used in adaptive Web agent systems before (Chen & Chen, 2002; Godoy & Amandi, 2000). When combined with the previous design goal this naturally leads to multiple, independent hierarchies.

4. **A user’s needs change over time:** In particular, needs may be short term, long term or periodic.

5. **Recent, active interests are most important:** The system should put the most attention to supporting the greatest needs and the current needs.

6. **The system should be capable of evaluating a page based on the user’s needs:** The needs in the hierarchies representing a user are each described by a weighted vector of keywords that can be used to evaluate a potential page. Each need can also refine its evaluation based on its parent need, having its expertise further specialized.

7. **The system should be able to search for pages that might be of interest to the user:** It is not enough to merely be able to model a user’s needs, but allow those needs to actively work on the user’s behalf.

8. **Like-minded users should share recommendations:** A user shares interests with the communities to which they belong,
and the SWAMI system is designed to take advantage of that idea. Elements of a user's representation can rub virtual shoulders with those from other users in a common location called a "rendezvous server."

9. **A Web site knows itself best:** The SWAMI architecture allows for interaction with expert agents which can be consulted for local, specialized recommendations, taking advantage of any hidden context they might have.

**High-Level Architecture**

SWAMI stands for "Searching the Web with Agents having Mobility and Intelligence." At the heart of SWAMI is a multi-agent system in which agents representing a front-end interface, the user's needs and page searchers. It is implemented using a (custom) multi-agent system (see Figure 1).

A user browses Web pages using the front-end interface, which passes the page to the multi-agent system representing the user's needs. The Web page is analysed and migrates to the most appropriate representative. At all times, the needs representation may be engaging search components to push recommendations back to the interface agent for the user to view.

**The Front-End Interface**

The user interacts with the system using the SWAMI interface agent (shown in Figure 2). This is currently integrated into a simple Web browser, allowing the user to interact with search and evaluations results, and allow the system to observe user activity.

This browser has basic features of a typical browser:

- It has a URL input field, so that a user may enter a URL and jump immediately to any page;
- It has history, and the user can navigate backward/forward through history;
- A link on any displayed page can be followed by clicking on it.

In addition, several specialized features have been added:

---

*Figure 1. A high level view of the SWAMI system*
Figure 2. A screenshot of the browser with integrated SWAMI

- There is a table of current page recommendations for the user to select from, if desired.
- The current analysis of any page can be displayed.
- Each of the agents working for the user can be inspected using the agent browser.
- The state of all agents can be saved and loaded, allowing the user to take a snapshot of the system.
- A message window at the bottom tells the user the current activity, including which agent won the bid for the current page, if a new agent has been created, retired or removed, and when a search has been conducted. (This was primarily used for monitoring activity during testing.)

**Representation of User Needs**

The user’s needs are represented by a hierarchically-arranged collection of agents. Each representation agent represents a cluster of pages the user has viewed. Agents are organized into hierarchies referred to as “corporations,” within which each represent an informal topic hierarchy devised from the agents themselves. Each corporation can be considered to represent a category of interest, where each agent is a specific topic within that category. (See Figure 3).

A similar approach to Godoy and Amandi (2002) has been taken in SWAMI, but with a significant difference: where in Godoy and Amandi (2002) an externally organized hierarchy was created, pages placed in that hierarchy, and the user’s interests being taken from a subset of that hierarchy, in SWAMI the hierarchy is developed entirely from scratch, allowing it to be a customized size to reflect the user’s interests.

**General Overview**

The hierarchies of agents are self-organized using an online, dynamic self-clustering technique. As pages are viewed by the user, an agent collects pages that are similar to the pages it has already gathered. The agent continually checks the compactness of its cluster, and if it is too loose (beyond a threshold), it will attempt to split the collection of pages up into tighter sub-
groups. If it is successful, it creates (or "hires") new agents to represent the subgroups. These agents are positioned below the original agent, so that incoming pages are first examined by the original agent, and then may be passed down to the more specialized sub-agents, and so on, until the best match has been made.

Initially, the interface agent collects all pages until a distinct group is discovered, forming the first representation agent. If no current agent is representative of a given page, the interface agent holds on to it until a new group manifests itself.

Definitions

In order to describe things more formally, a few definitions are needed, particularly with respect the notions surrounding clustering. Note that each agent represents a cluster of pages.

a. Document Cluster

A document cluster $C$ is a collection of documents that all share an acceptable level of similarity:

$$C = D_1, D_2, \ldots, D_n$$

(1)

$$\forall D_i, D_j \in C, \text{sim}(D_i, D_j) \geq \sigma$$

(2)

or at least the average similarity is above a threshold:

$$\frac{\sum \text{sim}(D_i, D_j)}{|C|} \geq \sigma$$

(3)

Ideally, that similarity is maximized for each document in this cluster:

$$\forall D_i \in C, C_i = \arg \max_{C_j \in C} \text{avgsim}(D_i, C_j)$$

(4)

where $C$ is the set of all clusters.

b. Cluster Centroid

To save time, a cluster may be represented by a set of features in much the same way as a document. This creates a "virtual" document (referred to as a centroid), which is the point with the most similarity (or least distance) from all other pages within the cluster:
\[
\forall D \in C, D_{\text{centroid}} = \arg \max_{D_j \in C, D_j \neq D} \text{angsim}(D, D_j)
\]  

(5)

Since each page is represented as a collection of terms, the ideal centroid may be described as the average of all pages, \(D\):

\[
\overline{D} = \left[ \sum_{D_i \in C} D_i \cdot w_1, \ldots, \sum_{D_i \in C} D_i \cdot w_n \right] / |C|
\]  

(6)

The set of all clusters being considered at a particular time is denoted by \(C\).

To denote that a document \(D\) is being considered in the context of a cluster \(C\), the notation \(D_{\overline{C}}\) is used.

**Page Representation**

The most important part of a Web page, from the perspective of the system, is the textual data it contains; specifically, the words with which the document is written with.

While tags sometimes indicate increased importance of a particular word within a document \(D\), initial investigation indicated that this importance is unpredictable, given the lack of strict adherence or interpretation of the use of tags to mark up specific words. Therefore, the frequency of a word occurring within a document regardless of mark-up is used as the basis for page description.

The interpretation of a word within a sentence is a complex problem requiring a sophisticated language-specific model. To increase the throughput of the system's analysis of Web pages and allow the system to be (largely) language-agnostic, sentence structure and punctuation is ignored for the processing of pages.

Thus, the basic representation is the well-known bag-of-words model. This representation collects all the unique words from a document and notes the frequency of each. The procedure for collecting these words involves eliminating HTML from the input, separating text from punctuation, reducing words to word stems, using the well-known Porter stemming algorithm (Porter, 1980), and removing arbitrarily uninteresting words listed in a stop list.

Once the words have been extracted from a document, it is necessary to determine which words are significant descriptors of the page, to extract the features from the words.

Features are terms which for which a sufficient weight has been calculated. For this purpose, the well-known TFIDF measurement was considered but has been modified to better suit the hierarchical context. In a hierarchical cluster, the terms of the parent cluster are already significant and have already been taken into account to form the parent cluster. To distinguish a child cluster and properly focus and specialize its terms, it is necessary to change the weighting scheme of the terms to take into account the fact that words from the parent are going to be present in all of the child cluster's pages.

The modified TFIDF-based calculation is as follows:

\[
w_i = f_i \times \left( \frac{DF_{C_p}(t)}{N_{C_p}} \right) \times \left( \frac{N_{C_c}}{DF_{C_p}(t)} \right)
\]  

(7)

where \(DF_{C_p}(t)\) is the document frequency of the term in the child cluster, \(C_p\) refers to the parent of cluster \(C_c\), \(DF_{C_p}(t)\) is the document frequency of the term in the parent cluster, \(N_{C_c}\) is the number of pages in the child cluster (including the candidate page), and \(N_{C_p}\) is the number of pages in the parent cluster.

If the cluster has no parent, the last term would result in a zero error, so the following truncated formula is used:

\[
w_i = f_i \times \left( \frac{DF_{C_c}(t)}{N_{C_c}} \right)
\]  

(8)

Note that this formula is nearly the opposite of the TFIDF formula. In TFIDF, a term is considered less important if it occurs in more
documents of a group, because the intent is to find terms that uniquely describe the page. In this case, terms that are common to more pages better describe the cluster, and are thus more important to the cluster. Those terms that are common to the cluster which are shared with the parent, however, are not as important to distinguish this cluster, because, by definition of being a subcluster of the parent, all of the parent’s terms are already found throughout the cluster; in other words, no new information is learned about a cluster by those words it inherits from its parent.

**Document Comparison**

Now that documents have been represented as a collection of weighted features, it is easier to compare documents against each other for similarity. A number of similarity measures were considered for this role, however, most are computation-intensive, and do not significantly improve upon the basic cosine similarity algorithm. The cosine method treats the features of documents as vectors in a multidimensional space, and calculates the angle between the vectors. As the angle decreases, the similarity between the two documents increases.

Note that in order for two documents to be properly compared, the feature weights need to be recalculated within the same context.

**Placement**

The placement of new pages is directly based on the ability to compare two pages discussed above.

To determine whether a document $D$ should be part of a cluster $C$, we can calculate the similarity between the document in the context of the cluster ($D_c$) and the cluster’s centroid, $D_c$:

$$\text{sim}(D_c, C) = \text{sim}(D_c, D_c)$$ (9)

The best cluster $\hat{C}$ for a given document $D$ is therefore the cluster that maximizes the similarity:

$$\hat{C} = \arg \max_{C \in c} \text{sim}(D_c, C)$$ (10)

Formally:

1. Given a set of existing clusters $C$ and a new document $D$, for each cluster $C_i$, recalculate the term weights of document $D$, producing a candidate document $D_{c_i}$.
2. For each candidate document $D_{c_i}$, calculate the similarity of the candidate to its cluster $C_i$:

$$\text{score}(D_{c_i}) = \text{sim}(D_{c_i}, D_{c_i})$$

3. A clear winning cluster $\hat{C}$ is discovered by finding a cluster for which the document score is better than every other cluster, and for which the score is greater than a given minimal similarity threshold ($\varepsilon = 0.6$):

$$C_i, \hat{C} \in C, \forall C_i \neq \hat{C}, \text{score}(D_{\hat{C}}) > \text{score}(D_{C_i}), \text{score}(D_{\hat{C}}) > \varepsilon$$ (11)

4. If there is a single winning cluster $\hat{C}$, the page is given to that cluster.
5. If there is a tie for the winning cluster or there is no clear winner, the page is held back in the “general area” until such time as it can be awarded to a clear winner.

**Hierarchical Placement**

When hierarchical clusters are involved, a distant descendant of a cluster may be the winning cluster. The simple placement algorithm can be extended by searching through all of the children for the best possible match. This can be accomplished by changing the score function

Copyright © 2009, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
of the cluster to return the best score between the cluster and its children:

\[
\text{score}(D_c) = \max \left[ \text{sim}(D_c, \overline{D_c}), \max_{\overline{D_c} \in C_c} \text{score}(D_c) \right]
\]  

(12)

Redistribution

As subclusters can change periodically, pages should be moved from a more general cluster to a more specific one whenever possible. The process of redistribution is the same as placement; the only difference being that the winning cluster must not only surpass all other clusters and the minimal similarity threshold, but also must be a better fit than the current cluster containing the page.

Splitting

Since pages are added incrementally and are held back if there is no clear winner, it is possible that a number of pages in a given cluster actually form a subcluster. The clustering program (in this case, an agent) must periodically examine the cluster it represents in order to discover these subclusters.

In order to find these subclusters, the following definition is used: given a set of documents \( D \), a subcluster is defined as a group of documents within that set for which the following holds:

1. The documents that best match/have the best similarity to a given document \( D' \) is denoted as the set \( B \subseteq D \);
2. All best match sets for each document within a given subcluster \( C \) are also contained within the subcluster;
3. All best match similarity scores are above a minimum similarity score.

If the set of documents \( D \) is viewed as a weighted graph, such that each vertex represents a document within that set, and each edge is weighted with the similarity score between the documents on its vertices, then the clusters are those distinct, non-overlapping subgraphs.

Given the pages \( D = \{A, B, C, D, E, F\} \) in the example in Figure 4, two subsets, \( \{A, B, C\} \) and \( \{D, E, F\} \) can be found.

Formally:

1. For all documents \( D_i, D_j \in D, D_i \neq D_j \), calculate the similarity score to be placed in the half-filled matrix \( M \). That is, each entry \( M[i][j] \) contains \( \text{sim}(D_i, D_j) \).
2. Initiate a set of clusters \( C \).
3. For each document \( D_i \):
   a. If \( \exists C \in C \) such that \( D_i \in C \), skip to the next document.
   b. Otherwise, find the set of best matching pages, \( B \); that is, those pages that have the best similarity scores with the given document. Together, this forms a potential cluster \( C' \).
   c. If any document in \( B \) is contained in any cluster of \( C \), this cluster is invalid.
   d. For each document \( D' \) in \( B \), repeat from step 3(b) until no new pages are discovered.
4. If the cluster is not of a sufficient size, the cluster is invalid.
5. If the cluster has not been invalidated, add the new cluster to the set of clusters.
6. Repeat from step 3.

Hierarchical Profile Agents

The representation of the user's needs is carried by Hierarchical Profile Agents. These agents are motivated by continued survival to organize Web pages on behalf of a user and find new pages related to those viewed. The current status of an agent is a measurement called "wealth."

Wealth Calculation
Each agent has a measurement of "wealth," which reflects the importance and relevance to the user of the cluster the agent represents.
Equation 13 shows the formula used to calculate weight. This combines the agent's size (sizeActivity), the success the agent has had in finding new pages for the user (search), the success the agent has had in having found pages accepted by the user (acceptance) and a history momentum which allows an agent to rest on its laurels briefly (wealth(t-1)).

\[
wealth(t) = \alpha \times sizeActivity + \beta \times search + \phi \times acceptance + \gamma \times wealth(t-1)
\]  

(13)

In this formula, \(\alpha\), \(\beta\), \(\phi\) and \(\gamma\) are arbitrary weights describing the relative importance of each factor. Useful values of each of these factors were discovered experimentally. A value of 0.25 was initially used, and was modified as the relative rates of change of each of the components were observed. The goal in this was to achieve an acceptable balance between the sustaining of active agents and the decay of inactive agents, while remaining sensitive enough to allow changes in the system to happen in a reasonable time frame. That is, the functioning of the system for each of the main points (agents increasing and decreasing in importance, agents splitting, agents retiring, agents rehiring and agents being removed) could be observed within a short number of page views.

After several iterations, the values were assigned as follows: \(\alpha=0.50\), \(\beta=0.25\), \(\phi=0.24\) and \(\gamma=0.01\). While these values were useful for demonstrating the system, in the future use of the system it is expected that they will have to be less sensitive.

Note that all of the factors add up to 1.0, and each of the component values has a range from between 0.0 and 1.0; this yields a weight between 0.0 and 1.0.

The agent's sizeActivity is a time-diminishing measure of the size of the agent balanced with how recently the agent was updated.

First, a few definitions: the size of an agent is calculated as the number of pages the agent holds and the total number of pages held by its children:
size(A) = |pages(A)| + \sum_{c \in \text{children}(A)} \text{size}(c) \tag{14}

The age of an agent is the number of pages it has bid on since its birth. The recency of activity is the difference between the current age and the age when a page was last added:

\text{recency}(A) = \text{age}(A) - \max_{p \in \text{page}(A)} \text{addAge}(p) \tag{15}

The general sizeActivity equation is:

\text{sizeActivity}(A) = \frac{f_1 + f_2}{2} \tag{16}

where \( f_1 \) and \( f_2 \) are defined in Equations 17 and 18:

\[
f_1 = \begin{cases} 
1.0 & \text{size}(A) > \text{age}(A) \\
\frac{\text{size}(A)}{\text{age}(A)} & \text{otherwise}
\end{cases}
\tag{17}
\]

\[
f_2 = \begin{cases} 
1.0 & \text{size}(A) = \text{recency}(A) = 0 \\
0 & \text{recency}(A) \geq \text{size}(A) \\
1.0 - \frac{\text{recency}(A)}{\text{size}(A)} & \text{otherwise}
\end{cases}
\tag{18}
\]

Two other special conditions apply: when the age of the agent is 0, the sizeActivity is 0, and when the size of the agent is zero but the recency is non-zero, the sizeActivity is 0.

The search success is a measure of how good recent searches have been, calculated as the proportion of good searches to all searches:

\[
\text{search}(A) = \frac{\sum_{r \in R_{\text{good}} \text{score}(r)} \left( \frac{\text{score}(r)}{\text{age}(A) - \text{birthAge}(r) + 2} \right)}{\sum_{r \in R} \left( \frac{\text{score}(r)}{\text{age}(A) - \text{birthAge}(r) + 2} \right)} \tag{19}
\]

where \( R \) is the set of all pending recommendations, \( r \) is an individual recommendation, \( \text{birthAge}(r) \) is the age of the agent when the recommendation was created, and \( R_{\text{good}} \) is the set of all good searches:

\[
R_{\text{good}} \subseteq R \forall r \in R_{\text{good}}, \text{score}(r) \geq \rho.
\]

The threshold \( \rho \) is arbitrarily set to 0.5 in the prototype.

Acceptance is a measure of how many recommendations were followed by the user, calculated as the proportion of recommendations followed relative to the number of recommendations made:

\[
\text{acceptance}(A) = \frac{|R_A|}{|R|} \tag{20}
\]

where \( R_A \subseteq R \), and all recommendations in \( R_A \) were followed by the user at some point.

Each of these formulae was created to arbitrarily represent elements believed to be of importance to an agent. They are all designed to diminish with a lack of activity over time.

Retirement

When an agent's wealth is reduced below a threshold, the agent is removed from the hierarchy and moved into a holding area ("retired"). In this way, agents which are not producing useful assistance are prunes from the hierarchy. However, to represent periodic interests, these agents are not immediately deleted. Rather, they remain in the holding area, continuing to decay, until one of two conditions is satisfied: either they are the best representative for a new page viewed by the user, or they represent a newly discovered subcluster better than a blank agent. In the first case, they become the head of a new corporation; in the second case they are simply added into the hierarchy at the appropriate point. This also allows subclusters to migrate to the most appropriate place; for example, a "Mexican cooking" agent might be retired from beneath the general "cooking" agent, but later
be rehired under a “Mexican culture” agent. (Note that agents are not labelled in this way; this is merely for illustration purposes.)

Life Cycle of an Agent
A hierarchical profile agent goes through the following steps in its life-cycle:

1. **Birth**: When a profile agent is first created, it is being created to take over the control of one or more pages. From this initial page collection it calculates its features to be used for page comparisons. The wealth of an agent after it has been initially created is the minimum wealth, 0.0.

2. **Page bidding**: Each time a user views a page, a profile agent is called to bid on that page. If the page is already owned by the agent or the page matches one of the agent’s recommendations, the value of the bid is 1.0 (perfect match). Otherwise, the value of a bid is simply the comparison between the page in the context of the agent’s pages and the agent’s features; in this case, the agent’s age increases by one, and its wealth is recalculated. Bids from all children agents \( C_i \) are also collected and returned.

3. **Page acceptance**: If a profile agent is the winner of a bid for a page, it will be given the page to own. If the page is one of the agent’s recommendations, that recommendation is marked as “accepted.” When a page is added, the agent is marked as increasingly “dirty.” When the dirtiness surpasses a threshold \( \delta \) (currently in the system, this is 2), the agent’s features are recalculated to match the possibly shifting centroid of its cluster. A page might not be accepted directly by an agent, but might be delivered to the appropriate winner of the bid.

4. **Page distribution**: If a page has been accepted by a child agent, that agent may have recalculated its features and pages owned by the parent agent might more appropriately fit with the child agent. Thus, the parent agent asks all of its child agents for a comparison score. Note that this is different from a bid, in that it does not increase the age of any agents. If there is a clear winner for a page, it is redistributed down the branch that won to the appropriate child agent. This allows a limited reconfiguration of the hierarchy, so that pages flow downward to the most specific agent for the page topics.

5. **Wealth recalculation**: If pages have been accepted or redistributed there will possibly be changes in the wealth of agents along the way.

6. **Subcluster search, or “splitting”**: At the core of the hierarchical structure is the ability of a cluster to contain one or more subclusters. An agent will search through its pages to attempt to find one or more maximal subclusters. For each of these subclusters a new child agent is created.

7. **Searching**: After a page has been accepted and the wealth recalculated, the agent may initiate a search if its wealth exceeds the threshold. The profile agent creates a new search agent of the appropriate type and starts that search agent in a separate thread. When that thread completes, the search agent will notify the profile agent that it has finished searching, and the profile agent will retrieve the list of recommendations from the search agent (if there are any) and destroy the search agent.

8. **Recommendation gathering**: At any time, the recommendations found for a particular profile agent may be requested. If the agent has not yet searched or is in the middle of searching, it may not have any recommendations. When the recommendations are requested of a hierarchical profile agent, it in turn requests all the recommendations from all of its children and collects them all in a list. If more than one agent has a recommendation for a particular page, the best recommendation is used.

9. **Retirement**: After an agent’s wealth has been recalculated, the agent may be deemed no longer relevant to the profile if its
wealth falls below a particular threshold, \( \eta \). When this happens, it will be removed from its current place (“retired”) and put into the holding area. The holding area is a special area maintained by the interface agent. This allows agents that performed well to exist longer, in case the topics that they represent are repeated in the future.

When in the holding area, an agent may not create any child agents nor create any search agents. It continues to make bids on incoming pages and therefore, to age. If it should win a bid from an incoming page, it will be reinstated from retirement and added as the head of a new corporation. This reflects the situation of a recurring interest, where the user had drifted away from an interest for a period, but has now returned to it.

If an agent fails to win any bids, its wealth will gradually decrease over time. Once its wealth has fallen below a second threshold, \( \tau \), it is permanently removed from the system.

Note that there is a “grace period” of 5 age units during which time an agent is not considered for retirement. This allows an agent to establish itself.

This stage allows interests that are no longer significant to fade away.

When a hierarchical profile agent is retired, all of the children agents become children of the agent’s parent agent. If the agent has no parent, that is, it is the head of a corporation, all of the child agents become heads of new corporations under the interface agent.

While retired, a hierarchical profile agent may not create any new children.

11. **Death:** If an agent falls below a lower threshold \( \tau \) of wealth, the agent is removed from the system. This only applies if the agent is already retired (and therefore has no children).

**Search Components**

When a representation agent reaches a sufficient level of wealth and experience, it can create search agents to work for it. Search agents take criteria from the representation agent (the set of word features the representation agent has used to form its cluster, for example) and attempts to find and evaluate pages on its behalf.

Four types of search agents have been considered for the system: agents who search the links from existing pages, agents which leverage search engines as a source of potential recommendations, agents which consult with local topic experts for recommendations, and agents which consult with other search agents.

**Link-following Search Agent**

The link-following search agent follows links from pages the user has already viewed and evaluates them based on its criteria. The agent, in a way, acts like a user in its pattern of browsing, following a similar depth-first pattern as discussed earlier.

**Search-engine based Search Agents**

The search-engine based search agent can submit different combinations of word features to a search engine and evaluate the results. In this way, it can take advantage of the massive database of knowledge available to a search
engine, but provide the personalization that the search engine lacks.

**Topic expert consulting Search Agents**
The topic expert consulting search agents are mobile agents which can travel to SWAMI-aware Web sites and interact with topic expert agents representing the Web page owner. These topic expert agents may have access to information that cannot be gathered from simply browsing the pages, and may be in a better position to provide recommendations. For example, the topic expert agents may know about arbitrary groupings of pages that do not have labels on the pages themselves.

**Collaborative Search Agents**
The collaborative search agent seeks to take advantage of the browsing behaviour of people with similar interests. It travels to a host (referred to as the “rendezvous server”) where it can interact with agents representing other people. There, they can swap recommendations based on how similar the agents are to each other. Another type of agent, the rendezvous hosts, remain in the rendezvous server at all times, interacting with all the visiting search agents and collecting all recommendations that they have. The rendezvous host becomes a “memory” for the rendezvous server, so that not all interactions between agents need to be synchronized.

**SUMMARY**

In order to provide personalized, ongoing, cross-Web site browsing navigation support to end-users, the SWAMI system was developed.

The SWAMI system contains a custom agent implementation written in Java. The majority of the agents form “corporations” representing user interests demonstrated from normal browsing behaviour, and map to clusters. An incremental, distributed clustering scheme is used to achieve this, and each agent is capable of independently initiating searches for other interesting Web pages on behalf of the end-user.

A number of parameters control the agent’s life cycle, including the balance of importance between agent size, agent searching, agent recommendation success and momentum. Other parameters include the retirement threshold and the death threshold of wealth. Parameters governing the clustering include the minimum number of pages an agent must see in order to search for clusters (the minimum dirtiness threshold).

**EVALUATION**

An initial evaluation of the SWAMI system consisted of developing a prototype and determining if it was capable of detecting multiple, different interests by observing browsing behaviour, and of then acting on behalf of the test user to search out and evaluate new potential pages of interest.

To perform this evaluation, test data was generated that represents a web of pages that are interconnected and that have localized coherence. From this test data, numerous trial runs were conducted in order to demonstrate that all of the key events expected of the system were observed, and that the system was behaving as expected. Four typical example runs are highlighted here.

Because the test data was generated offline and never made available to a search engine, no examination of the search-engine-based search method could be attempted, without implementing a specialized search engine, which was beyond the scope of this initial research.

**Test Methodology**

The SWAMI system was tested with a test end-user by selecting a random start page and then having that end user follow links according to one of five typical user motivations, derived from the earlier observations about browsing behaviour:
1. **Continued interest**: Links might be chosen to represent a continued interest in an existing topic. The links followed for this motivation were links to pages within groups from which other pages had been visited in this session.

2. **Changing interest**: Links might be chosen to represent a break in the interests of the user. These links were chosen at random from the page being viewed or their URLs were entered directly into the input area.

3. **Returning interest**: Links might be chosen that represent a return to a past interest. Often, these links were chosen to be relevant to an agent that had been retired but not yet deleted from the system, to demonstrate that such an agent would be returned to active service.

4. **Intense and specialized interest**: Links might be chosen to represent a deepening and possibly specialized interest within an existing topic. These links were chosen from the best recommendations provided by the system.

5. **Shallow interest**: Links might be chosen to represent the seeking behaviour of a user, wherein links are essentially chosen at random.

Test runs were continued until it had demonstrated a particular feature. In total, 25 test runs were complete.

For each test run, the following events were observed:

- When an agent is created;
- When an agent splits; this is distinct from the previous event in that it concentrates on the effect of the change on the parent, rather than the details of the new child agent;
- The bids for each page from each agent in the system;
- The wealth of each agent after each bid
- When a search was conducted, and the results it found;
- When an agent is retired;
- When an agent is reinstated;
- When an agent is deleted permanently.

To summarize the results, two charts were devised: the **page group activity** chart and the **agent activity** chart. The time scale on both of these charts is in terms of page views by the user.

The **page group activity** chart shows the activity of the user in terms of the pre-generated page group to which each page the user chose belonged. Note that pages within the same pre-generated page group share a high similarity, while pages between groups tend to have a much lower similarity, so a steady line indicates a consistent interest that the system should recognize.

The **agent activity** chart shows the wealth of all of the agents in the system over the same period. As agents accumulate pages and become more certain about the topics they represent, their wealth should increase; as the topics they represent fall out of favour, the wealth should decrease. The identification of subtopics can also be seen in the creation of child agents, which is shown as a sudden start of a weight track on the chart. When agents are removed, their weight track disappears.

These two charts, when taken together, show the inputs and outputs of the system, and the better they correspond, the better the system is tracking the user and acting on their behalf.

**Results**

The following sections describe the results from two particular trial runs in detail. These trial runs were chosen to clearly illustrate the performance of the system, but are typical of the results.

It should be noted that the system uses a particular naming scheme for all agents within the system. All agents begin with the prefix “Charlie” and have a suffix indicating their parentage. In the case of the topmost agent, the Interface Agent, no suffix is used. The first child of the Interface Agent is called “Charlie_0”;
the second child of this agent would be called “Charlie_0_1”, and so on.

To describe the life-cycle of an agent, a chart showing the agents’ wealth over time is used. This chart is calibrated in absolute terms, meaning that while an individual’s age is calculated relative to when they were born, it has been adjusted to the appropriate real outside age relative to the age of the Interface Agent. The age also describes the number of unique pages viewed. Where a line begins on the graph indicates when an agent was born; if the line ends prematurely, that agent was removed from the system.

The lower threshold for an active agent’s wealth before being retired is 0.2; only the Interface Agent cannot be retired. If their wealth continues to drop, an agent will be removed when it falls below 0.15.

Pages within a particular group are known to be similar to each other, and thus represent a topic. This is used both to train the system and to interpret its results.

Also, as the agents search, they discover pages in other page groups that are relevant to the topic, thus forming a virtual topic group based on the user’s demonstrated interests.

Example 1: Interest Shifts

This example demonstrates SWAMI’s ability to follow a user’s changing interests and react accordingly. The page groups that the user visited can be seen in Figure 5. On the weight track in Figure 6, five agents (in addition to the Interface Agent) are shown.

Each agent was created when the system detected a cluster of similar pages. The set of pages initially chosen all came from pre-generated group 40, followed by a number of pages selected from group 38. Charlie_0 was created when the subset of pages from group 40 were detected as distinct, at age 4. At age 11, a second agent (Charlie_1) was created to take control of the second subcluster discovered (for group 38). Note that while the pages were chosen from the pre-generated group, the system itself has no knowledge of these groups.

Between ages 14-36, links were followed semi-randomly from existing pages, but not corresponding to any previous page. These pages were similar enough to existing agents that Charlie_0 rose in wealth during this time period, and Charlie_1 maintained a high wealth. Concentration by the user on a single pre-generated page group again from age 36-43 resulted in the creation of a new agent, Charlie_2, to handle a newly-discovered cluster formed out of those pages. Another agent, Charlie_3 was created at the same time, as the new pages highlighted some previous cluster in the previous pages.

Between ages 53 and 77, recommendations made by Charlie_0 were followed, resulting in that agent’s consistent wealth, while other agents diminished. At age 77, a new topic was focused on, and a new agent, Charlie_4 was created in response.

Note that when the user concentrated on a particular topic, the system responded by creating a new agent to handle this new topic when it detected it. As the user drifted away from that topic (by not visiting again), the agents that had been responsible for it waned in wealth.

The longevity of both Charlie_0 and Charlie_1 indicate long-term interests. Charlie_0, in particular, has received a lot of attention from having suggestions followed.

Charlie_2 and Charlie_3 accurately map to short-term interests. In the case of Charlie_3, no recommended pages from that agent were viewed, leading it to degrade in wealth very quickly and disappear within about 5 page views. Charlie_2 was a short-term interest which the user paid a little attention to.

Finally, Charlie_4 is a new interest to which the user is paying attention and good recommendations have been found. The system responds quickly to the newly discovered cluster, and it becomes the most influential among them.

This example has shown that the system creates new agents to handle new user interests, and the wealth of those agents reflects the ongoing interest in the topic they represent.
Figure 5. The page groups visited by the user on the first example test run

Figure 6. Agent activity from the first example test run

Example 2: Interest Specialization

In stark contrast to the previous example, this example demonstrates the creation of specialized agents for sub-topics discovered within the context of a larger topic. While the page group activity shown in Figure 7 seems to be chaotic (particularly after age 57), the corresponding location on Figure 8 shows relatively stable behaviour.
Charlie_0 represents a long-term interest (page group 42) which was concentrated on for a considerable period. Two sub-topics were detected from within this one, represented by Charlie_0_0 and Charlie_0_1. The second of these was pursued momentarily, but was forgotten.
for a period. Note that Charlie_0_1 was retired but brought back instantaneously when the user returned to that topic. At that point, it actually triggered a split, creating the very short term topic represented by Charlie_0_1_0.

At approximately age 45, the system has detected that the user has decided to view another topic intensely for which good suggestions could be found. This is represented by Charlie_1, whose continued strength is due to its suggestions being followed. The return of a peak in Charlie_0 at approximately age 72 was due to following a link on a page suggested by Charlie_1 which led off to an older topic.

The relative stability of the wealth track of Charlie_1 after age 45 despite the apparent randomness of the page group activity for the same time period is due to the agent having found pages within multiple groups which are similar to the topic at hand. In this way, it has created a virtual group of pages centred around the user’s interests.

CONCLUSION

This article describes a framework for a multi-agent system for providing personalized Web page recommendations to users. The SWAMI framework features a sophisticated user model using a social multi-agent system with a cost-driven and time-variable interaction model organized into hierarchies of related topics. Agents representing particular topical interests in this system can search for recommendations for the user with one of multiple strategies. Among those search strategies is the ability of the search agents to become mobile. Mobile search agents can travel to particular, SWAMI-aware Websites and interact with local topic experts, or they can travel to SWAMI “rendezvous servers,” where they can interact with user-independent collaborative recommendation agents and with other search agents representing users.

Key features of this framework include local representation of a user’s interests (allowing the system to “learn once, apply everywhere”), the integration of local, site-based and collaborative recommendations, and an active user profile representation which takes into account short-term, long-term and recurring interests, as well as the specialization of interests.

This holistic approach to Web search represents a more realistic solution to the problem of Web search than site-specific or user-agnostic approaches.

Several trial runs were performed, from which typical examples were chosen to examine in detail. These trial runs demonstrate that the agents do grow to mirror the user activities and change over time to reflect changes in the user intentions. Short-term, long-term and recurring interests have been detected by the system, as well as specialization to accommodate a particularly important interest. Recommendations could be gathered successfully by using a link-search algorithm, by consulting with site experts or through interacting with a community. Recommendations in the community were successfully distributed between members of that community.

Future Work

This research has proved promising, but there are several additional questions raised throughout the work that would make interesting follow-up research. These questions include:

- **Parameter setting**: There are a large number of parameters within this system, including: minimum similarity thresholds; minimum agent wealth before retirement; minimum agent wealth before deletion; the relative weights of each component in the agent wealth calculation; the minimum number of pages required before an agent considers splitting; the minimum number of pages that must remain in a cluster after all others have been allocated to sub-clusters; and so on. These parameters are currently established through observation and arbitrary decision. However, in some cases, these parameters do not always work correctly. It seems natural that these parameters might either be
adjusted by evidence or even determined entirely by evidence within the system. In addition, it may also be beneficial to allow parameters to be tuned according to user tastes.

- **Agent hierarchy/structure reorganization:** As agents are retired and rehired, the hierarchical structure of the agents is reorganized, allowing multiple independent but similar specializations the possibility of converging in one branch. This specialization will only occur, however, when interest in a particular sub-agent has waned significantly. There seems to be a natural role for a “headhunter” agent or something similar which can help facilitate reorganizations of the agent structure without the need of a diminished interest.

- **Open agent structure:** The agents are currently arranged in a strictly hierarchical manner, with each agent having at most one parent and any number of children. This structure, while convenient, is artificial; information often does not follow a strictly hierarchical structure, instead having more of an open graph structure. One possible modification of the SWAMI architecture is to modify the concept of “parent” and “child” agents into the more general “ancestor” and “descendant” roles, or even into the most general “sibling” role. Such a structural change would be capable of modelling much more subtle interactions within the data, but each agent would have a larger web of information from which to discover patterns.

- **Page comparison:** Several page comparison mechanisms were examined before selecting the cosine similarity measure. In particular, several variations on the Jaccard measure were strong contenders. Only a few of the measures examined take into account structural or positional information about the term on a page. Alternative methods of page comparison might improve the identification of page clusters and the ability for pages to reach the appropriate agent. Pages are also currently viewed only as the collection of terms that physically occur within the body of the document (in their stemmed forms). The system might be significantly improved if a facility such as WordNet could be included to search for other words based on common relationships such as antonym or synonym, although at a cost of complication and processing time.

- **Incremental calculation:** When a page is added to a new cluster, it affects the centroid of that cluster, which may shift the set of features enough that some of the pages that had been part of the cluster may no longer fit properly within it, or might fit more appropriately within a sub-cluster. Thus, every time a page is added to a cluster, a significant amount of re-calculation is potentially performed, significantly impacting performance. If it could be possible to calculate only the effect itself on the cluster instead of calculating everything, or to calculate a predictor that can indicate whether a full recalculation should be done, this would significantly improve the speed of the system.

- **Local expert agents:** Only one form of the local expert agent was examined in the process of this thesis, but several are possible. In particular, it might be possible for the local experts to reorganize themselves to reflect the kind of requests that are being made, making the local experts adaptive to usage.

- **Rendezvous agents environment:** The rendezvous server implemented here is basically functional, but it does not have any sophisticated mechanisms for creating new rendezvous host agents. Currently, if no host agents are discovered to serve a particular incoming search agent, a new host agent is created. Also, there is little attempt to guarantee that two host agents do not end up covering the same
material; in fact, in the evaluation of the rendezvous server, there were in fact two host agents that had a difference of only two pages.

- **Remote search environment locating:** The current system looks for a remote search environment (whether rendezvous server or local expert environment) on the local machine, at a known port. To make the system more generally useful, a mechanism for identifying these remote environments is necessary.

- **Choosing search methods:** In the current system, the method of searching is hard-wired into the particular incarnation of the executable. It is desirable that multiple search methods be available simultaneously to the system. It is also desirable to allow the agents to choose which method is appropriate, perhaps under the direction or suggestion of the user. This includes preferences, perhaps, for different remote search environments for different topics. It is expected that the search methods would complement each other well: local expert agents allow specialized exploration in a particular local environment; link search agents allow easy exploration between local page environments; rendezvous servers link individuals together into communities, allow transfer of recommendations between users; and the search-engine-leveraging search agents allow disconnected local environments to be connected.

- **User control and manipulation of agents:** It has been speculated that if users were able to tag or name the agents working for them, they would be able to further judge the recommendations provided by those agents. Other controls might also be useful, such as the ability to freeze an agent from changing (thus always providing the same kind of judgement without fear of being retired), arbitrarily remove an agent (to prune the system), or arbitrarily reward an agent for a particularly useful suggestion.

- **Page re-occurrence:** The system currently takes a simplified view of pages: they are unchanging entities, so when a page has been viewed once, it need never be viewed again. The interface tracks the list of pages that have been viewed so far this session, and simply does not process those pages that have already been seen. This simplification works for a large number of pages, but a significant portion of the Web changes constantly. If these changing pages could be identified, the system could automatically scan the pages to see if they have changed enough to be revisited by the user.

- **User profile persistence:** Currently, the system is only designed to work for the duration of a single session. The ability to save the state of the system was experimentally tried with early in the development, but it was vulnerable to incompatibilities introduced by code changes. For such a system to be generally useful, however, it must have a way to store a user’s profile between sessions. In a similar way, the rendezvous server should have the ability to persistently store the community it represents.

- **Wealth as accumulated value:** The system calculates the wealth of an agent as an instantaneous value based on the performance and other history of the agent. An alternative view is to treat wealth more like the real economic concept, in that it is something acquired for success and paid out to perform actions or to simply “live.” The instantaneous system was implemented to give some measure of control and confidence that the system would have a continuing downward trend if no beneficial activity took place. A more open economy also requires additional controls to ensure that it changes appropriately and maintains a good balance.

- **Earlier detection of groups:** To circumvent the “cascading group” problem, the system requires that a small number of pages be left behind in a parent agent.
before a child agent can be created. In particular, the interface agent, which is not capable of searching itself, will not create any corporations if there are not a sufficient number of pages that remain after the corporation has been created. This leads to the system apparently not able to find a group until after the user has left it, because only by visiting pages that are inconsistent with the previous pages can a new group be formed (leaving those most recent pages behind).

- **Real user testing:** This system was tested with a model of real users over pages which have known properties. The next stage of testing will involve real users and a much broader set of pages with more variable qualities, such as with the general Web. Preliminary testing in this manner has yielded promising results. In more open testing, users would be able to rate the search results discovered and provide a qualitative score to both determine the real success of the search agents and to provide feedback for the system to choose which avenues of search are most fruitful.

## REFERENCES


Mark Kilfoil received master’s degree in computer science from The University of New Brunswick, Canada, in 2004, and is currently pursuing his PhD degree. He is a founding member of the Intelligent and Adaptive Systems Research Group at UNB. His current research interests include multi-agent negotiation, adaptive web systems, mobile agents and information management and filtering technologies.

Ali Ghorbani has held a variety of positions in academia for the past 29 years including heading up project and research groups and as department chair, director of computing services, director of extended learning and as assistant dean. He received his PhD and master’s in computer science from the University of New Brunswick, and the George Washington University, Washington D.C., USA, respectively. Ghorbani currently serves as dean of the Faculty of Computer Science. He holds UNB research scholar position. His current research focus is Web intelligence, network & information security, complex adaptive systems, and critical infrastructure protection. He authored more than 220 reports and research papers in journals and conference proceedings and has edited 8 volumes. He served as general chair and program chair/co-chair for 7 International Conferences, and organized over 10 International Workshops. Dr. Ghorbani is the founding director of Information Security Centre of Excellence at UNB. He is also the coordinator of the Privacy, Security and Trust (PST) network at UNB. Ghorbani is the co-editor-in-chief of Computational Intelligence, an international journal, and associate editor of the International Journal of Information Technology and Web Engineering and the ISC journal of Information Security.