A Context Ultra-Sensitive Approach to High Quality Web Recommendations based on Web Usage Mining and Neural Network Committees

Olfa Nasraoui and Mrudula Pavuluri
Dept. of Electrical and Computer Engineering
206 Engineering Science Bldg.
The University of Memphis
Memphis, TN 38152-3180
onasraou@memphis.edu

ABSTRACT

Personalization tailors a user's interaction with the Web information space based on information gathered about them. Declarative user information such as manually entered profiles continue to raise privacy concerns and are neither scalable nor flexible in the face of very active dynamic Web sites and changing user trends and interests. One way to deal with this problem is through a completely automated Web personalization system. Such a system can be based on Web usage mining to discover Web usage profiles, followed by a recommendation system that can respond to the users' individual interests. We present several architectures that rely on pre-discovered user profiles: *Context Sensitive Approaches based on single-step Recommender systems (CSA-1-step-Rec)*, and *Context Ultra-Sensitive Approaches based on two-step Recommender systems (CUSA-2-step-Rec)*. In particular, the two-step recommendation strategy based on a committee of profile-specific URL-Predictor models, is more accurate and faster to train because only the URLs that are relevant to a specific profile are used to define the relevant attributes for this profile's specialized URL-Predictor model. Hence, the model complexity, such as the neural network architecture, can be significantly reduced compared to a single global model that could involve hundreds of thousands of URLs/items. The two-step approach can also be expected to handle *overlap* in user interests, and even to mend the effects of a profile dichotomy that is too coarse. Finally, we note that all the mass-profile based recommendation strategies investigated are intuitive, and are low in recommendation-time cost compared to collaborative filtering, (no need to store or compare to a large number of instances). In our simulations on real Web activity data, the proposed context ultra-sensitive two-step recommendation strategy achieves unprecedented high coverage and precision compared to other approaches such as K-NN collaborative filtering and single-step recommenders such as the Nearest-Profile recommender.

Keywords
Web personalization, web recommendation, web usage mining, neural networks, collaborative filtering.

1. INTRODUCTION

The flow of information in a completely automated Web personalization system spans several stages starting from the user's Web navigation patterns until the final recommendations, and including the intermediate stages of logging the users' activities, preprocessing and segmenting Web log data into Web user sessions, and learning a usage model from this data. The usage model can come in many forms: from the lazy type modeling used in collaborative filtering[15,16], that simply stores all the users' information and then relies on K Nearest Neighbors to provide recommendations from the previous history of neighbors or similar users; to a set of frequent itemsets or associations; to a set of clusters of user sessions, and resulting Web user profiles or summaries. Pazzani and Billsus [7] presented a collaborative filtering approach to recommendation, based on users' ratings of specific web pages, and Naives Bayes as the prediction tool. Mobasher et al. [9] use pre-discovered association rules and an efficient data structure to provide faster recommendations based on web navigation patterns. Collaborative models do not perform well in the face of very sparse data, and do not scale well to the huge number of users and URLs/items [15]. In association rule based methods, large support thresholds yield only a very small fraction of the total number of itemsets, while smaller support thresholds generally result in a staggering number of mostly spurious URL itemsets. Techniques that are based on clustering to discover profiles are also expected to face serious limitations in the presence of huge, very high-dimensional, and sparse usage data, unless the clustering technique is efficient, robust to noise, and can handle the sparseness. Unsupervised robust multi-resolution clustering techniques [6] can reliably discover most of the Web user profiles/interest groups, because they benefit from a *multi-resolution* mechanism to discover even the less pronounced user profiles, and they benefit from *robustness* to avoid the noisy/spurious profiles that are common with low support thresholds. Their *unsupervised* nature also avoids reliance on input parameters or prior knowledge about the *number* of profiles to be sought. Linden et al. [14] used *item-to-item collaborative filtering* as a recommendation strategy on *Amazon.com®*. This approach matches each of the user's purchased and rated items to

similar items, then combines those similar items into a recommendation list. A *similar-item table* is built by finding items that customers tend to purchase together.

In this paper, we propose several *Context Sensitive Approaches based on single-step Recommender systems (CSA-1-step-Rec)*, and *Context Ultra-Sensitive Approaches based on two-step Recommender systems (CUSA-2-step-Rec)*. The single-step recommender systems *(CSA-1-step-Rec)* simply predict the URLs that are part of the nearest estimated profile as recommendations. For example, the Nearest-Profile prediction model simply bases its recommendations on the closest profile based on a similarity measure. While this is a very simple and fast approach, it makes the critical assumption that sessions in different profiles are linearly separated. While this may be applicable for certain web mining methods, it may not be true for others. In order to be able to reliably map new unseen sessions to a set of mined profiles, without such assumptions about the profiles or how they separate the sessions, we can resort to classification methods that can handle non-separable profile boundaries. In this paper, we explore both decision trees and Multi-Layer Perceptron neural networks for this task. Once trained, using the decision tree or neural network model to classify a new session is fast, and constitutes the single step of the recommendation process, since the classified profile *is* the recommendation set.

The Context Ultra-Sensitive two-step recommender system *(CUSA-2-step-Rec)* first maps a user session to one of the pre-discovered profiles, and then uses one of several profile-specific URL-predictor neural networks (such as Multilayer perceptron or Hopfield Autoassociative memory networks) in the second step to provide the final recommendations. Based on this classification, a different recommendation model is designed for each profile separately. A specific neural network was trained offline for each profile in order to provide a profile-specific recommendation strategy that predicts web pages of interest to the user depending on their profile and the current URLs. The two-step recommendation method not only handles *overlap* in user interests, but can mend the effects of misclassification in the first nearest profile assignment step, and even the effect of a coarse profile dichotomy. The two-step recommendation method based on multilayer perceptron networks achieves unprecedented high coverage and precision compared to K-NN and Nearest-Profile recommendations. However, the approach based on Hopfield Autoassociative memory networks seemes to struggle against the well known harsh constraints limiting the number of patterns that can be memorized relative to the number of units/URLs. Finally, we note that the proposed recommendations are intuitive, and low in cost compared to collaborative filtering: They are faster and require lower main memory at recommendation time (no need to store or compare to a large number of instances).

The rest of the paper is organized as follows. In Section 2, we present an overview of profile discovery using Web usage mining. In Section 3, we present the single-step profile prediction based recommendation process, and the two-step recommender system based on a committee of profile-specific URL-predictor neural networks. In Section 4, we present an empirical evaluation of the recommendation strategies on real web usage data, and finally, in Section 5, we present our conclusions.

2. PROFILE DISCOVERY BASED ON WEB USAGE MINING

The first step in intelligent Web personalization is the automatic identification of user profiles. This constitutes the *knowledge discovery engine*. These profiles are later used to recommend relevant URLs to old and new anonymous users of a Web site. This constitutes the *recommendation engine*, and constitutes the main focus of this paper.

The knowledge discovery part based on *Web usage mining* [2,3,6,17,18,19,20] can be executed *offline* by periodically mining *new* contents of the user access log files. In this paper, Web usage mining is used to discover the profiles using the following steps:

(1) Preprocess log file to extract user *sessions*,
(2) *Categorize* sessions by *Hierarchical Unsupervised Niche Clustering (H-UNC)* [6],

(3) Summarize the session categories in terms of *user profiles*,
(4) Infer *context-sensitive* URL *associations* from user profiles.

Step 1: Preprocessing the Web Log File to extract User Sessions

The access log of a Web server is a record of all files (URLs) accessed by users on a Web site. Each log entry consists of the following information components: *access time, IP address, URL viewed, ...etc*. An example showing two entries is displayed below

17:11:48 141.225.195.29 GET /graphics/horizon.jpg 200
17:11:48 141.225.195.29 GET /people/faculty/nasraoui/index.html 200

The first step in preprocessing [1,2,3] consists of mapping the N_U URLs on a website to distinct indices. A user session consists of accesses originating from the same IP address within a predefined time period. Each URL in the site is assigned a unique number $j \in 1, \ldots, N_U$, where N_U is the total number of valid URLs. Thus, the ith user session is encoded as an N_U-dimensional binary attribute vector $s^{(i)}$ with the property

$$s_j^{(i)} = \begin{cases} 1 & \text{if user accessed } j^{th} \text{ URL} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Step 2: Clustering Sessions into an Optimal Number of Categories

For this task, we use *Hierarchical Unsupervised Niche Clustering* [6] or *H-UNC*. *H-UNC* is a *hierarchical* version of a *robust genetic* clustering approach (*UNC*) [5]. A Genetic Algorithm (GA) [14,15,16,17,18] evolves a population of candidate solutions through generations of competition and reproduction until convergence to *one* solution. Hence, the GA cannot maintain population *diversity*. *Niching* methods attempt to maintain a *diverse* population with members distributed among *niches* corresponding to multiple solutions. An initial population of randomly selected sessions is coded into binary chromosome strings that compete based on a *density* fitness measure that is

highest at the centers of good (dense) clusters. Different niches in the fitness landscape correspond to distinct clusters in the data set. The algorithms are detailed below. Note that the clusters *and their number* are determined automatically. More details on HUNC can be found in [6].

Hierarchical Unsupervised Niche Clustering Algorithm (*H-UNC*) [6]:
-Encode binary session vectors
-Set current resolution Level $L = 1$
-Start by applying *UNC* to entire data set w/ small population size;
-Repeat recursively until cluster cardinality or scale become too small {
  -Increment resolution level: $L = L + 1$
  -For each parent cluster found at Level (L-1):
    -Reapply *UNC* [5] only on data subset assigned to this parent cluster
    -Extract more child clusters at higher resolution ($L > 1$)
}

Step 3: Summarizing Session Clusters into User Profiles

After automatically grouping sessions into different clusters, we summarize the session categories in terms of *user profile vectors* [3,4], $\boldsymbol{p}_i$**:** The k^{th} component/weight of this vector ($\boldsymbol{p}_{ik}$) captures the *relevance* of URL_k in the i^{th} profile, as estimated by the conditional probability that URL_k is accessed in a session belonging to the i^{th} cluster (this is the frequency with which URL_k was accessed in the sessions belonging to the i^{th} cluster). The model is further extended to a *robust profile* [3,6] based on robust weights that assign *only* sessions with high robust weight to a cluster's *core*. Unpolluted by noisy (irrelevant) sessions, these profiles give a *cleaner* description of the user interests.

3. DESCRIPTION OF THE SINGLE-STEP AND TWO-STEP RECOMMENDATION STRATEGY OPTIONS

Let $U = \{\text{url}_1, \text{url}_2, \ldots, \text{url}_{N_U}\}$ be a set of N_U urls on a given web site visited in web user sessions $\boldsymbol{s}_j$, $j = 1, \ldots, N_s$, as defined in (1). Let $P = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{N_P}\}$ be the set of N_P Web user profiles computed by the profile discovery engine. Each profile consists of a set of URLs associated with their relevance weights in that profile, and can be viewed as a relevance vector of length N_U, with p_{ik} = relevance of url_k in the i^{th} profile. The problem of recommendation can be stated as follows. Given a current Web user session vector, $\boldsymbol{s}_j = [s_{j1}, s_{j2}, \ldots, s_{jN_U}]$, predict the set of URLs that are most relevant according to the user's interest, and recommend them to the user, usually as a set of Hypertext *links* dynamically appended to the contents of the Web document returned in response to the most recent Web query. Because the degree of relevance of the URLs that are determined of interest to the user, may vary, it may also be useful to associate the k^{th} recommended URL with a corresponding URL relevance *score*, r_{jk}. Hence it is practical to denote the recommendations for current Web user session, $\boldsymbol{s}_j$, by a vector $\boldsymbol{r}_j = [r_{j1}, r_{j2}, \ldots, r_{jN_U}]$. In this study, we limit the scores to be binary.

3.1 Context Sensitive Approach Based on Single-Step Profile Prediction Recommender System (*CSA-1-step-Rec*)

3.1.1 Single-Step Nearest-Profile Prediction Based Recommender System

The simplest and most rudimentary approach to profile based Web recommendation is to simply determine the most similar profile to the current session, and to recommend the URLs in this profile, together with their URL relevance weights as URL recommendation scores.

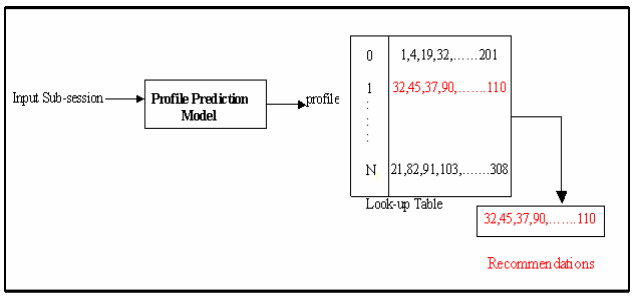

Figure 1: Context-Sensitive Approach based onn single-step profile prediction based Recommender System *(CSA-1-step-Rec)*. The Profile Prediction Model can be a Nearest-Profile classifier or any of the models shown in Figs 2 or 3.

Figure 1 shows the structure of such a recommendation system, where the profile prediction model simply consists of a nearest-profile estimator based on computing a session to profile similarity, and selecting the profile with highest similarity as the predicted profile.

The similarity score between an input session, $\boldsymbol{s}$, and the i^{th} profile, $\boldsymbol{p}_i$, can be computed using the cosine similarity as follows,

$$S_{si}^{cosine} = \frac{\sum_{k=1}^{N_U} p_{ik} s_k}{\sqrt{\sum_{k=1}^{N_U} p_{ik} \sum_{k=1}^{N_U} s_k}} \qquad (2)$$

If a hierarchical Web site structure should be taken into account, then a modification of the cosine similarity, introduced in [3,4], that can take into account the Website structure can be used to yield the following input membership,

$$S_{si}^{web} = \max\left\{ \frac{\sum_{l=1}^{N_U}\sum_{k=1}^{N_U} p_{il} S_u(l,k) s_k}{\sum_{k=1}^{N_U} p_{ik} \sum_{k=1}^{N_U} s_k}, S_{si}^{cosine} \right\} \qquad (3)$$

where $\boldsymbol{S_u}$ is a URL to URL similarity matrix that is computed based on the amount of overlap between the paths leading from the root of the website (main page) to any two URLs, and is given by $S_u(i,j) = \min\left(1, \frac{|p_i \cap p_j|}{\max(1, \max(|p_i|, |p_j|) - 1)}\right)$

We refer to the special similarity in (3) as the *Web Session Similarity*.

3.1.2 Single-Step Decision-Tree Based Profile Prediction Recommender System

The nearest profile prediction model makes the critical assumption that sessions in different profiles are linearly separated. While this may be applicable for certain web mining methods, it may not be true for others. In order to be able to reliably map new unseen sessions to a set of mined profiles, without such assumptions about the profiles or how they separate the sessions, we can resort to classification methods that are not based on distance or similarity computations. In this paper, we explore both decision trees and neural networks for this task. Once trained, using the decision tree or neural network model to classify a new session is fast, and constitutes the single step of the recommendation process, since the classified profile *is* the recommendation set.

The decision tree profile prediction model is very similar to the nearest profile prediction model. An input binary vector is presented as input to the decision tree [22] and a profile/class is predicted as the output. Each URL in the input vector is considered as an attribute. In learning, first the entire training data set is presented. Here, an attribute value is tested at each decision node with two possible outcomes of the test, a branch and a sub-tree. The class node indicates the class to be predicted.

Building the tree [22] starts by selecting the URL with maximum information gain, and placing this URL, say *url1* at the root node. The test at the branch is whether the session data includes a visit to *url1*. If so, they are placed as one group. The next URL with the maximum gain is placed as the node of the sub-tree. For instance, let the next URL be *url2*. Again all the remaining sessions are checked to see if they traversed *url2*. If yes, they are sub-grouped. This procedure is continued until all the attributes are checked or all the class attributes are identical (a pure sub-sample). This example is illustrated in fIgure 2.

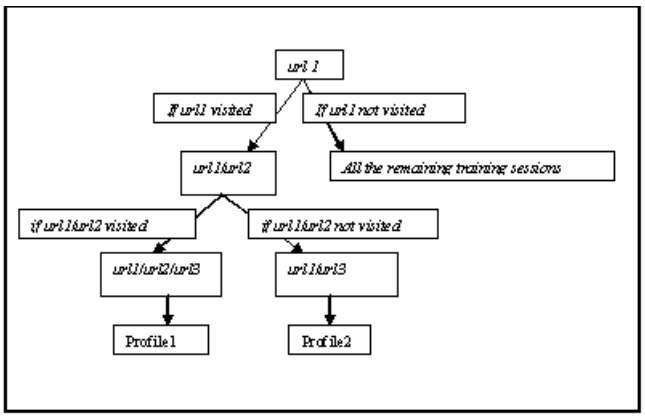

Figure 2: Example of a Profile Prediction Model based on a decision tree that can be used within *CSA-1-step-Rec*

3.1.3 Single-Step Neural Network Based Profile Prediction Recommender System

In the neural network [21] based approach of profile prediction, a feed-forward multilayer perceptron is used and is trained with Back-Propagation. The inputs (session URLs) and output (class or profile) to the prediction model remain the same as the ones described above. The neural network replaces the classification model block in Figure 1. Hence the input layer of the network consists of as many input nodes as the number of valid URLs (i.e. N_U nodes), an output layer having one output node for each profile (i.e. N_p nodes), and a hidden layer with $(N_U + N_p)/2$ nodes. Figure 3 shows the architecture of the neural network used to predict the most relevant profile. The index of the output node with highest activation indicates the final class/profile.

During the learning process, the input is passed forward to the other layers and the output of each element is computed layer by layer. The difference between the acquired output of the output layer and the desired output is back propagated to the previous layers (modified by a derivative of the transfer function) and the weights are adjusted. Learning of the network continues until all the weights are learnt and no further change in the weights occur.

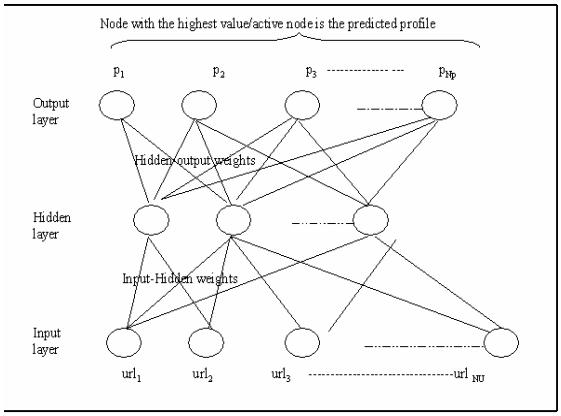

Figure 3: Architecture of a Profile Prediction Model based on a Multi-Layer Perceptron that can be used within *CSA-1-step-Rec*

3.2 Context Ultra-Sensitive Approach Based on Two-Step Recommender System with A Committee Of Profile-Specific URL-Predictor Neural Networks (*CUSA-2-step-Rec*)

The single-step Profile prediction recommendation procedure is intuitively appealing and very simple. In particular, its implementation and deployment in a live setting is very efficient. Essentially, it amounts to a look-up table. However, it has several flaws: ***(i)*** the degree of similarity between the current session and the nearest profile that is identified may not be taken into account, ***(ii)*** the above procedure does not take into account sessions that are similar to more than a single profile, ***(iii)*** it cannot handle sessions which are different from all known profiles, and ***(iv)*** the set of recommendations derive directly from the contents of a single (assigned) profile for all sessions assigned to this profile, without any further distinction between the specific access patterns. For this reason, we propose a two-step approach that in addition to exploiting the profile information, is able to recommend more highly personalized recommendations that depend not only on the assigned profile (*people-to-people* collaboration filtering), but also explicitly, on the input session itself (*item-to-item* collaboration filtering),.

In this method, the URLs reflecting the more specific user interests, which *may differ even between users assigned to the*

same profile, are predicted as recommendations instead of the same profile. Hence, the more individualized URL predictions are expected to perform better compared to the coarser single-step profile prediction model.

3.2.1 Description of the Multi-Layer Perceptron URL-Predictor Neural Network

A Multilayer Perceptron neural network [21] can be used for directly predicting the URLs to be given as recommendations. The architecture of the network is different from the network used in the profile prediction scenario of Figure 3, and is shown in Figure 4. This is because the number of output nodes is now equal to the number of input nodes. Each training input consists of a user sub-session (*ss*) derived from a ground-truth complete session *S*, while the output nodes should conform to the remainder of this session (*S-ss*). This means that there is one output node per URL. Hence, the architecture of the network can become extremely complex, as there would be N_U input and N_U output nodes. This will in turn increase the number of hidden nodes to roughly N_U nodes. Training such a network may prove to be unrealistic on large websites that may consist of thousands of URLs. To overcome this problem, a separate network is learned for each profile independently, with an architecture of its own. Here, the number of input and output nodes depends only on the number of significant URLs in that profile, and possibly those related to its URLs by URL-level or conceptual similarity, and the number of hidden nodes is set to the average of number of input and output nodes. Figure 4 shows the architecture of each URL-predictor neural network. There will be a committee of N_p specialized networks of similar kind used in developing this URL recommendation prediction model, as illustrated in Figure 5. Each of these networks is completely specialized to forming the recommendations for only one profile, hence offering a local, more refined model, that enjoys the advantages of better accuracy, simplicity (fewer nodes and connections), and ease of training (as a result of simplicity).

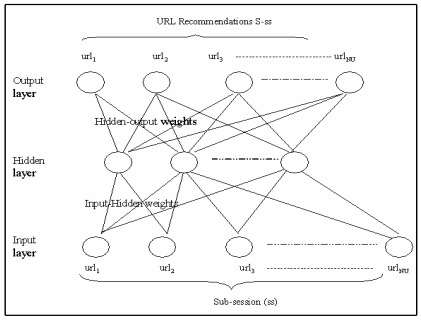

Figure 4: Architecture of a Profile-Specific URL-Predictor Neural Network used in *CUSA-2-step-Rec*

3.2.2 Learning the Profile-Specific URL-Predictor Neural Network Models

The URL-Predictor network for each profile is learnt independently with a separate set of training data. Learning each network involves presenting a *sub-session* consisting of some of the URLs visited by the user belonging to that profile as input and adjusting the network weights by *back propagation* to recommend URLs that are not part of the sub-session given as input, but which are a part of the ground truth complete session, as output of the network. For each ground truth complete session, we find all the sub-sessions for window sizes 1-10, and use them to generate independent training and testing sets. Cosine similarity is used to map each sub-session to the closest profile, and the URL-Predictor network specialized for that profile is invoked to obtain the recommendations. Hence the input/output units of the URL-Predictor network specialized for a particular profile are limited to only the URLs that are present in the sessions assigned to that profile. For this reason, we re-index the URLs in the sub-sessions to a smaller index set, so that the number of input nodes in each network will be reduced compared to the total number of URLs in the website. This will reduce the complexity of the network's architecture. The output URLs of the invoked network are given as the URL *recommendations* to the user. The output URL recommendations (*R*) are in the binary format and are decoded into the URLs by applying a threshold of *'0.5'*. That is, the URL is considered to be recommended if its activation value exceeds a *'0.5'* at the corresponding output node of the network.

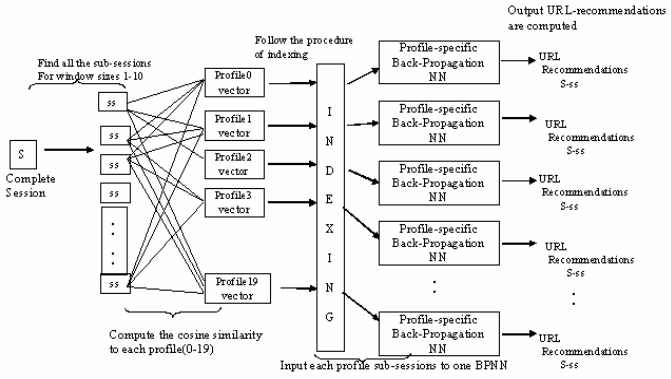

Figure 5: Context Ultra-Sensitive Approach based on Two-Step Recommendation Process *(CUSA-2-step-Rec)* using a Committee of Profile-Specific URL-Predictor Neural Networks (Any URL-Predictor model can be substituted for the Multi-Layer Perceptron, e.g. a Hopfield network)

3.3 Recommendations Based On Autoassociative Memory Hopfield Networks

Hopfield networks are a special kind of recurrent neural networks that can be used as associative memory [21]. A Hopfield network can retrieve a complete pattern stored through the training process from an imperfect or noisy version of it. In some sense, a recommender system performs a similar operation, when it recommends certain URLs from an incomplete session. Given N_{url} fully connected (via symmetric weights w_{ij} between each two units i and j) neurons, each serving simultaneously as input and as output, and assuming that the activation values, x_i, are bipolar (+1/-1), the optimal weights to memorize N_p patterns, can be determined by *Hebbian* learning as follows

$$w_{ij} = \sum_{p=1}^{N_p} x_i^p x_j^p \text{ for all } i \neq j \text{ (0, otherwise)}$$

During testing/recall, when a new noisy pattern x_{new} is presented as input, we set the activation at node i at iteration 0 to be $x_i^0 =$

$x_{new\text{-}i}$, then the units are adjusted by iteratively computing, at each iteration t

$$x_i^{t+1} = \sum_{j=1}^{N_{url}} w_{ij} x_j^t$$

until the network converges to a stable state. However, the desired behavior of recall in a Hopfield network is expected to hold only if all the possible complete session prototypes can be stored in the Hopfield network's connection weights, and if these complete sessions do not interact (or *cross-talk*) excessively. Severe deterioration starts occurring when the number of patterns $N_p > 0.15N_{url}$, hence limiting Hopfield recommender system to sites with a large number of URLs and yet very little variety in the user access patterns. This limitation is paradoxical in the context of large websites or transactional database systems. Our preliminary simulations with both a single global Hopfield network as well as several profile-specific Hopfield networks have resulted in low recall qualities since the network seemed to be able to memorize only very few stable states. However several profile-specific Hopfield networks perform better than one global network, but only for some of the profiles.

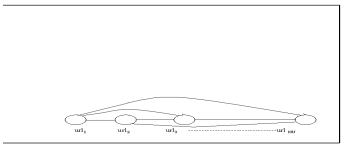

Figure 6: Architecture of an Auto-Associative Memory Hopfield URL-Predictor Neural Network that can be used in *CUSA-2-step-Rec*

3.4 Evaluating the Recommender Systems

First a data set consisting of real web user sessions extracted from the log files of a Web server is used to generate training and testing sets. For each complete session considered as the ground-truth, all the possible sub-sessions of different sizes are generated up to 9 URLs. If a user session consists of more than 10 URLs, then we first randomly select 10 URLs and then use them as the basis to form all sub-sessions. This is done in order to maintain the nesting or inclusion relationship between sub-sessions of size (k) and sub-sessions of size ($k+1$). This in turn will facilitate the interpretation of trends in performance versus the sub-session size.

Once the URL-prediction model is built, we need to evaluate its performance by presenting test sub-sessions to the trained network. The test dataset is generated similarly to the training dataset, but it forms an independent 20% of the complete set of sub-sessions generated.

Each actual completed session, $\boldsymbol{s}_T$, is treated as ground-truth, a subset of this session is treated as incomplete current sub-session, $\boldsymbol{s}_j$, and the computed recommendations are treated as predicted complete session. This process is very similar to an information retrieval problem. Hence the evaluation proceeds by computing the *precision* and the *coverage* of the recommendations.

Let $r^*_j = r_j - \boldsymbol{s}_j$. This corresponds to the recommendations obtained after omitting all URLs that are part of the current subsession. Also, let $\boldsymbol{s}^*_j = \boldsymbol{s}_T - \boldsymbol{s}_j$, be the set of ground truth URLs, not including the ones in the current subsession being processed for recommendations. The recommendation precision is given by

$$prec_j = \frac{\sum_k^{N_U} r^*_{jk} s^*_{jk}}{\sum_k^{N_U} \left(r^*_{jk}\right)^2} \tag{4}$$

and coverage is given by

$$\text{cov}_j = \frac{\sum_k^{N_U} r^*_{jk} s^*_{jk}}{\sum_k^{N_U} \left(s^*_{jk}\right)^2} \tag{5}$$

In most recommender systems, precision and coverage act in contradictory fashion. That is while one increases, the other decreases, and vice versa. To take into account both of these measures, a measure known as the "effectiveness" or F1-Measure can be computed. This measure achieves higher value when both the precision and coverage are simultaneously high. F1 is given by

$$F1 = \frac{2 * \text{Pr}ecision * Coverage}{\text{Pr}ecision + Coverage} \tag{6}$$

4. EXPERIMENTAL RESULTS

4.1 Mining User profiles from Anonymous Web Usage Data

Hierarchical Unsupervised Niche Clustering (H-UNC) [6] was applied on a set of web sessions preprocessed from the 12 day access log data of the Web site of the department of Computer Engineering and Computer Sciences at the University of Missouri, Columbia. After filtering out irrelevant entries, the data was segmented into 1703 sessions accessing 343 distinct URLs. The maximum elapsed time between two consecutive accesses in the same session was set to 45 minutes. We applied H-UNC to the Web sessions using a maximal number of levels, $L = 5$, in the hierarchy, and the following parameters that control the final resolution: $N_{\text{split}} = 30$, and $\sigma_{\text{split}} = 0.1$. H-UNC partitioned the Web users sessions into 20 clusters at level 5, and each cluster was characterized by one of the profile vectors, $\boldsymbol{p}_i$, i=0, ...,19. Some of these profiles are summarized in Table 1.

TABLE 1. SOME OF THE 20 DISCOVERED PROFILES

No.	size	Relevant URLs/Profile	Profile Description
0	106	{0.29 - /staff.html} {0.92 - /} {0.98 - /people.html} {0.99 - /people_index.html} {0.97 - /faculty.html} {0.15 - /degrees.html} {0.20 - /research.html} {0.35 - /grad_people.html} {0.26 - /undergrad_people.html}	Main page, people, faculty, staff, degrees, research.
1	104	{0.99 - /} {1.00 - /cecs_computer.class}	Main page only
2		{0.81 - /} {0.75 - /cecs_computer.class} {0.87 - /courses.html} {0.90 - /courses_index.html} {0.88 - /courses100.html} {0.22 - /courses300.html} {0.16 - /courses_webpg.html} {0.16 - /~lan/cecs353}	Main page and course pages
3	61	{0.80 - /} {0.48 - /degrees.html} {0.23 - /degrees_grad.html} {0.23 - /degrees_grad_index.html}	Main page and *graduate* degrees

		{0.23 - /deg_grad_genor.html}	
4	58	{0.72 - /} {0.97 - /degrees_undergrad.html} {0.95 - /degrees_index.html} {0.97 - /bsce.html} {0.52 - /bscs.html} {0.34 - /bacs.html} {0.34 - /courses.html} {0.34-/courses100.html} {0.26 - /general.html} {0.22- /courses300.html}{0.81 - /degrees.html} {0.19 - /courses200.html}	Main page, *undergraduate* courses, degrees, especially *undergraduate* degrees.
13	38	{0.47 - /~shi} {0.82 - /~shi/cecs345} {0.21 - /~shi/cecs345/java_examples} {0.34 - /~shi/cecs345/references.html}	CECS345 Java examples/references

4.2 Comparative Simulation Results for *CUSA-2-step-Rec*, *CUSA-2-step-Rec*, and *K-NN*

We used the following parameters in training the multilayer perceptron URL-Predictor neural networks: Maximum number of epochs = 2000, Learning Rate = 0.7 (for Input to Hidden layer) and 0.07 (for Hidden to Output layer). This parameter is important in learning the network as it gives the amount of change of weights at each step. These optimal values were selected by trial and error, because with a very small value, it may take a long time for the algorithm to converge; while a large value will make the algorithm diverge by bouncing around the error surface. The learning rate is set higher for the connections from input to hidden layer when compared to the one set for hidden to output layer. This makes the learning slower and more accurate for the output layer nodes. We have also used a Momentum factor of 0.5, to accelerate learning while reducing unstable behavior.

The Collaborative filtering approach is based on using K Nearest Neighbors (K-NN) followed by top-N recommendations for different values of K and N. First the closest K complete sessions from the entire history of accesses are found. Then the URLs present in these top K sessions are sorted in decreasing order of their frequency, and the top N URLs are treated as the recommendation set. We show only the best results obtained for K-NN at K=50 neighbors and N=10 URLs.

Figures 7 through 9, depicting the 20-profile averaged measures (precision, coverage, and F1), show that the two-step profile-specific URL-predictor multilayer perceptron neural network recommender system *(CUSA-2-step-Rec)* wins in terms of both better precision and better coverage, particularly above session size 2. This appeared at first unusual since it is very rare that a recommendation strategy scores this high on both precision and coverage, and that an increase in precision did not seem to compromise coverage in any way. However, by looking at the details of the design of the profile-specific URL-predictor neural network, we explain this relentless increase in precision by the fact that the neural network output is trained to predict only the URLs that the user has *not* seen before, i.e. '*S-ss'*, where *S* is the *complete* session, and *ss* is the sub-session (URLs *visited* by the user). Clearly, as the sub-session size increases, more URLs are presented to the output of the Neural network, making the prediction task easier, since fewer URLs need to be predicted compared to smaller input sub-sessions. Similarly, coverage increases, since with more input URLs, the neural network is able to predict more of the missing URLs to complete the puzzle. However, this does not happen at the expense of precision. On the contrary, in this special type of URL prediction neural network, giving more hints about the user in the form of more of the visited URLs makes the prediction task easier, and hence, will only result in more accurate predictions.

We notice that the single-step recommender systems *(CSA-1-step-Rec)* do not have this nice feature, i.e., precision and coverage will generally have opposing trends. However the single-step neural network profile prediction recommender seems to have a rather increasing precision, unlike the other single-step methods (nearest profile and decision trees). Again, this may be attributed to the fact that the more sophisticated neural network succeeds in learning the true profile/class better regardless of the session size.

The performance of k-NN fares competitively with all the single-step recommender strategies, but only for longer session sizes. This is not surprising, considering that k-NN can yield very accurate predictions, because it too is based on local context-sensitive models. However, k-NN is notorious for its *excessive computational and memory costs, at recommendation time*, in contrast to all the other investigated techniques. While lazy in the learning phase, involving nothing more than storing all the previously seen cases, k-NN takes its toll during the recommendation phase, when it needs to compare a new session with all the historic cases in order to produce recommendations.

Based on the F1 measure, shown in Figure 9, that combines both precision and coverage equally, we can conclude that the single-step recommender systems vary in performance depending on the range of session sizes. The single-step Nearest Profile predictor recommender system works best for very small session sizes (less than 2). The single-step Decision Tree predictor recommender system works best for session sizes in the range 3-5, and then competes very closely with the single-step Neural Network predictor recommender system above session size 5. While k-NN outperforms the single-step recommender systems, the two-step profile-specific URL-predictor neural network recommender system wins in terms of both better precision and better coverage, particularly above session size 2. This suggests using a combination of different recommender modules that are alternated based on session size, even within the lifetime of a single user session.

Finally Fig. 10 shows the averaged F1 measure values for the sessions belonging to each profile separately, when the *CUSA-2-step-Rec* strategy is used. The label for each profile is indicated close to its own curve. We notice that performance for most profiles is very good, except for profile 3. By looking at Table 1, profile 1 is seen to grab some URLs with low URL significance weight. This confirms the presence of some sessions, that while sharing the main profile URLs, also have additional URLs that altogether are accessed very infrequently. These additional URLs are very hard to model in the URL prediction network, because the overwhelming majority of the sessions in this profile do not contain them. In fact, using the robust session identification approach described in [6], we were able to identify that only half of the sessions in profile 3 form a strong consensus of two URLs. The remaining sessions cannot be modeled effectively even with a highly specialized URL prediction model. Since we did not screen these widely varying noise sessions from the recommendation process, their effect will be seen. Exploiting the robustness feature of HUNC [6] is an interesting and worthwhile step towards greatly enhancing the quality of the recommendations, and we leave this open for future investigation.

Fig. 11 shows the averaged F1 measure values for the sessions belonging to each profile separately, when the *K-NN* strategy is used. We can see a more sporadic behavior of the quality of the recommendations for different sizes, witnessing to the inadequacy of a global (i.e. across *all* profiles) collaborative

neighborhood model for some profiles and session sizes, especially in such a non-metric, noisy, and sparse high dimensional space. Note that when the measure curves dive to zero at a certain session size, this means that there were no sessions of longer size, and does not reflect the quality of recommendations.

Most importantly, we notice that the widely varying performance for different profiles suggests that different recommendation strategies perform best for different profiles, and it may be beneficial to combine different strategies depending not only on the current session size, as we have noticed previously, but also depending on the identified profile, yielding a sophisticated hybrid meta-recommendation system.

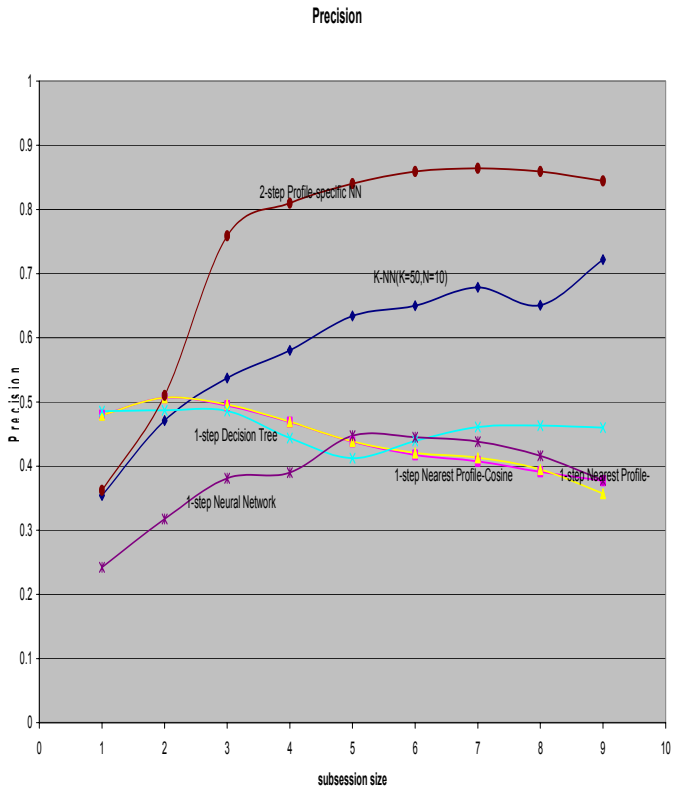

Figure 7: Precision Values for all recommendation strategies (*CSA-1-step-Rec*, *CUSA-2-step-Rec*, and *K-NN*)

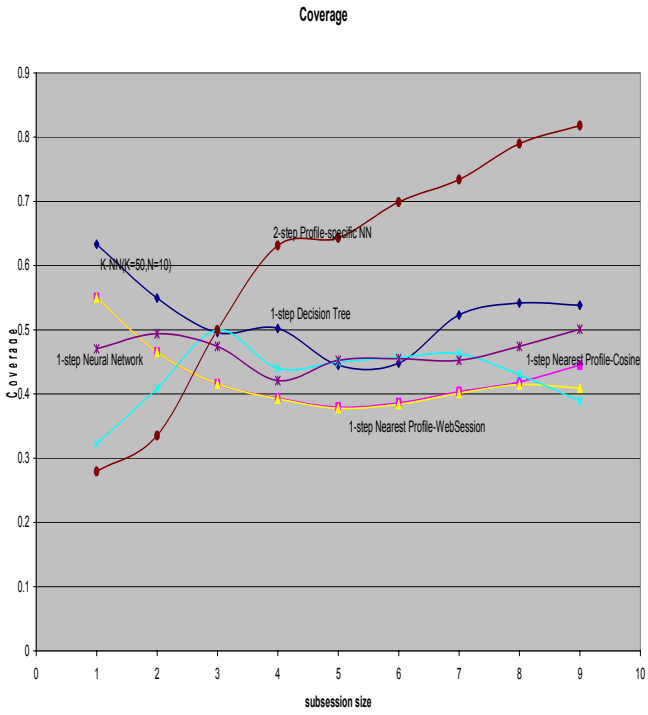

Figure 8: Coverage Values for all recommendation strategies (*CSA-1-step-Rec*, *CUSA-2-step-Rec*, and *K-NN*)

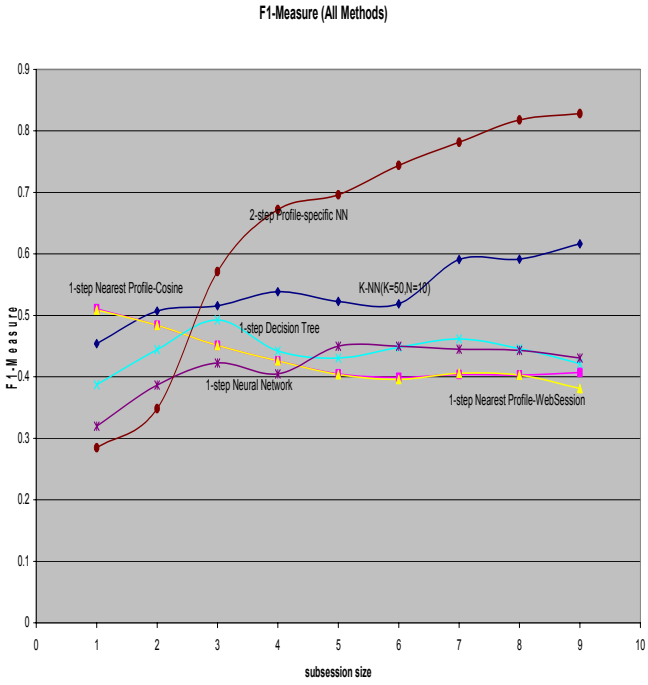

Figure 9: F1-Measure Values for all recommendation strategies (*CSA-1-step-Rec*, *CUSA-2-step-Rec*, and *K-NN*)

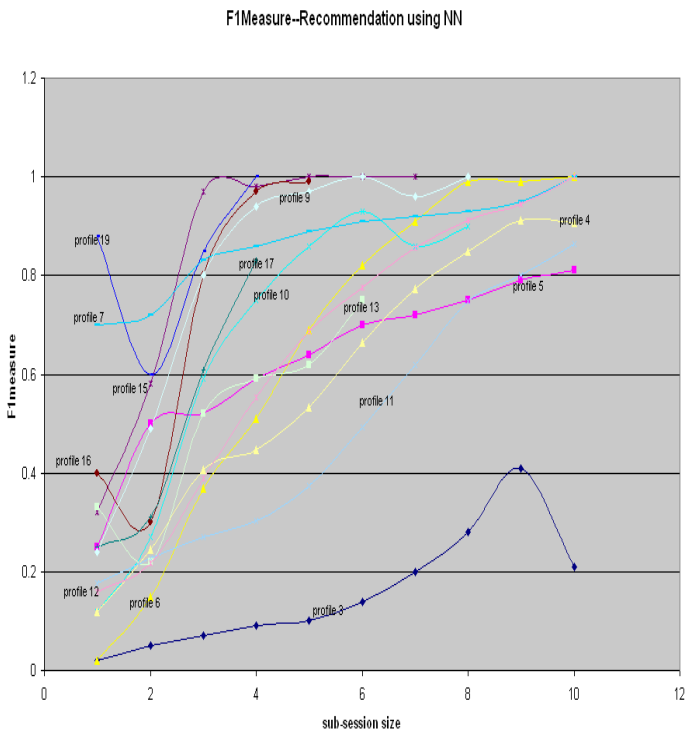

Figure 10: Individual averaged F1-Measure Values for each profile with the *CUSA-2-step-Rec* strategy

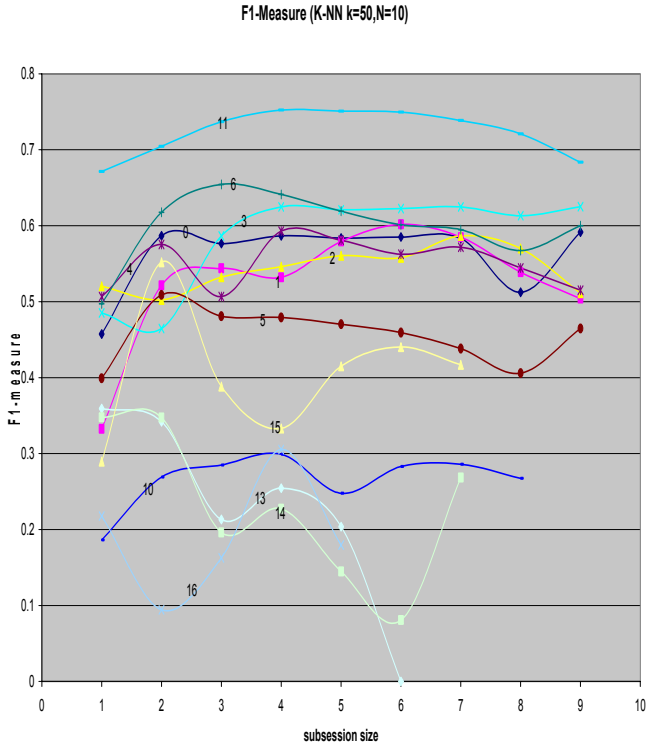

Figure 11: Individual averaged F1-Measure Values for each profile with the *K-NN* (K = 50, N = 10) strategy

4.3 Time and Memory Complexity Considerations

From the point of view of cost or time and memory complexity, it is interesting to note that *training* may take longer with the profile based approaches. However this training can be done on a back end computer and not on the server, and is therefore an *offline* process that does not affect the operation of the web server. On the other hand, all profile based approaches are extremely fast at *recommendation time* and require a minimal amount of main memory to function since they work with a mere summary of the previous usage history instead of the entire history as in k-NN collaborative filtering [15]. In our simulations, the proposed profile-based recommendations with non-optimized Java code running on a 1.7 GHz *Pentium* IV PC took only a small fraction of a second to provide one recommendation, resulting in the order of 10-100 recommendations per second depending on the strategy used[1]. The far more computationally complex K-Nearest Neighbor collaborative filtering generated recommendations at a leisurely 2 recommendations per second.

5. CONCLUSIONS

We have investigated several single-step and two-step recommender systems. The single-step recommender systems *(CSA-1-step-Rec)* simply predict the URLs that are part of the nearest estimated profile as recommendations. The nearest profile prediction model simply based its recommendations on the closest profile based on a similarity measure. In order to be able to reliably map new unseen sessions to a set of mined profiles, without such assumptions about the profiles or how they separate the sessions, we can resort to more powerful classification methods. In this paper, we explored both decision trees and neural networks for this task. Once trained, using the decision tree or neural network model to classify a new session constitutes the single step of the recommendation process, since the classified profile *is* the recommendation set.

The two-step recommender system *(CUSA-2-step-Rec)* first maps a user session to one of the pre-discovered profiles, and then uses one of several profile-specific URL-predictor neural networks in the second step to provide the final recommendations. Based on this classification, a different recommendation model is designed for each profile separately. A specific back-propagation neural network was trained offline for each profile in order to provide a profile-specific recommendation strategy that predicts web pages of interest to the user depending on their profile. The two-step recommendation method achieves unprecedented high coverage and precision compared to K-NN and single-step nearest-profile recommendations. Finally, we note that the proposed recommendations are low in cost compared to collaborative filtering: They are faster and require lower main memory at recommendation time (no need to store or compare to a large number of instances). Even though training neural networks to

[1] Here, 1 *recommendation per second* is actually a full set of recommendations for a given session and *not* individual URLs per second.

build a recommendation system may take a long time *offline*, *testing* this built network on a new user may take *just a small fraction of a second*.

Unlike most previous work, the proposed two-step profile-specific URL-predictor neural network recommender system allows a more refined context sensitive recommendation process. The idea of using a separate network specialized to each profile seems to be novel, since it provides an even higher level of context-awareness in personalization than the level already offered through collaborative filtering based personalization. The proposed method *simultaneously* achieved precision and coverage values of unprecedented quality (higher than 0.6 for most session sizes). Most existing techniques tend to excel in one measure (such as precision), and fail in the other (example, coverage).

It is important to note that the divide-and-conquer strategy adopted in the 2-step approach to break what would be an unmanageable learning task (URL prediction for all sessions regardless of their profile) into several simpler learning tasks (each profile separately) was the key to both **(i)** enabling faster learning, **(ii)** enabling better performance by reducing interactions between different sessions. It is reasonable to expect that this modular design could be extended by replacing the URL-Predictor neural network modules by different learning paradigms that are faster to train, while not compromising the accuracy of predictions. Such learning modules could be decision trees. The proposed model could also be made even faster to train and more accurate by encouraging the discovery of even more high-resolution profiles, which is specifically one of the desirable features of Hierarchical Unsupervised Niche Clustering.

We finally classify our recommendation approaches with respect to the two-dimensional taxonomy presented in [16]. First, because the user is *anonymous* at all times, our approaches are all *ephemeral* with respect to the *persistence dimension*. Second, with respect to the *automation dimension*, our approaches are *fully automatic*. Furthermore, with regard to the four different families of recommendation techniques identified in [16] (*non-personalized, attribute based, item-to-item correlation, and people-to-people correlation*), the 1-step recommenders (*CSA-1-step-Rec*) can be considered as people-to people collaborative filtering. However, they use a cluster/profile summarization model, hence providing better scalability. On the other hand, the *CUSA-2-step-Rec* model uses people-to people collaborative filtering that is summarized through a cluster model, in the first stage to map a new user to a profile. Then it uses a specialized item-to-item recommendation model to produce the final recommendations. Hence the *CUSA-2-step-Rec* approach can be considered as a *hybrid* between *people-to-people* and *item-to-item* recommendations, which may explain its superior performance.

Acknowledgements

This work is supported by National Science Foundation CAREER Award IIS-0133948 to O. Nasraoui.

References

[1] M. Perkowitz and O. Etzioni. Adaptive web sites: Automatically learning for user access pattern. Proc. 6th int. WWW conference, 1997.

[2] R. Cooley, B. Mobasher, and J. Srivastava, Web Mining: Information and Pattern discovery on the World Wide Web, Proc. IEEE Intl. Conf. Tools with AI, Newport Beach, CA, pp. 558-567, 1997.

[3] O. Nasraoui and R. Krishnapuram, and A. Joshi. Mining Web Access Logs Using a Relational Clustering Algorithm Based on a Robust Estimator, 8th International World Wide Web Conference, Toronto, pp. 40-41, 1999.

[4] O. Nasraoui, R. Krishnapuram, H. Frigui, and A. Joshi. Extracting Web User Profiles Using Relational Competitive Fuzzy Clustering, International Journal on Artificial Intelligence Tools, Vol. 9, No. 4, pp. 509-526, 2000.

[5] O. Nasraoui, and R. Krishnapuram, A Novel Approach to Unsupervised Robust Clustering using Genetic Niching, Proc. of the 9th IEEE International Conf. on Fuzzy Systems, San Antonio, TX, May 2000, pp. 170-175.

[6] O. Nasraoui and R. Krishnapuram. A New Evolutionary Approach to Web Usage and Context Sensitive Associations Mining, International Journal on Computational Intelligence and Applications - Special Issue on Internet Intelligent Systems, Vol. 2, No. 3, pp. 339-348, Sep. 2002.

[7] M. Pazzani and D. Billsus, Learning and revising User Profiles: The identification of Interesting Web Sites, Machine Learning, Arlington, 27, pp. 313-331, 1997.

[8] Kraft, D.H., Chen, J., Martin-Bautista, M.J., and Vila, M.A., Textual Information Retrieval with User Profiles Using Fuzzy Clustering and Inferencing, in Szczepaniak, P.S., Segovia, J., Kacprzyk, J., and Zadeh, L.A. (eds.), Intelligent Exploration of the Web, Heidelberg, Germany: Physica-Verlag, 2002.

[9] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, Effective personalizaton based on association rule discovery from Web usage data, ACM Workshop on Web information and data management, Atlanta, GA, Nov. 2001.

[10] J. H. Holland. *Adaptation in natural and artificial systems*. MIT Press, 1975.

[11] L. Zadeh (1965). Fuzzy sets. Inf. Control 8, 338-353.

[12] Klir, G. J., and Yuan, B., Fuzzy Sets and Fuzzy Logic, Prentice Hall, 1995, ISBN 0-13-101171-5.

[13] R. Agrawal and R. Srikant (1994), Fast algorithms for mining association rules, Proceedings of the 20th VLDB Conference, Santiago, Chile, pp. 487-499.

[14] G. Linden, B. Smith, and J. York, *Amazon.com* Recommendations Iem-to-item collaborative filtering, IEEE Internet Computing, Vo. 7, No. 1, pp. 76-80, Jan. 2003

[15] J. Breese, H. Heckerman, and C. Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, Proc. 14th Conf. Uncertainty in Artificial Intelligence, pp. 43-52, 1998.

[16] J.B. Schafer, J. Konstan, and J. Reidel, Recommender Systems in E-Commerce, Proc. ACM Conf. E-commerce, pp. 158-166, 1999.

[17] J. Srivastava, R. Cooley, M. Deshpande, and P-N Tan, Web usage mining: Discovery and applications of usage patterns from web data, SIGKDD Explorations, Vol. 1, No. 2, Jan 2000, pp. 1-12.

[18] O. Zaiane, M. Xin, and J. Han, Discovering web access patterns and trends by applying OLAP and data mining technology on web logs, in "Advances in Digital Libraries", 1998, Santa Barbara, CA, pp. 19-29.

[19] M. Spiliopoulou and L. C. Faulstich, WUM: A Web utilization Miner, in Proceedings of EDBT workshop WebDB98, Valencia, Spain, 1999.

[20] J. Borges and M. Levene,Data Mining of User Navigation Patterns, in *"Web Usage Analysis and User Profiling",* Lecture Notes in Computer Science", H. A. Abbass, R. A. Sarker, and C.S. Newton Eds., Springer-Verlag , pp. 92-111,1999.

[21] S. Haykin, *Neural* Networks: A Comprehensive Foundation, Macmillan, New York, 1994.

[22] J. R. Quinlan. Induction of Decision Trees. Machine Learning, Vol. 1, pp. 81--106, 1986.