Effective Personalization Based on Association Rule Discovery from Web Usage Data

Bamshad Mobasher, Honghua Dai, Tao Luo, Miki Nakagawa
{mobasher,hdai,tluo,miki}@cs.depaul.edu
School of Computer Science, Telecommunication, and Information Systems
DePaul University, Chicago, Illinois, USA

Abstract

To engage visitors to a Web site at a very early stage (i.e., before registration or authentication), personalization tools must rely primarily on clickstream data captured in Web server logs. The lack of explicit user ratings as well as the sparse nature and the large volume of data in such a setting poses serious challenges to standard collaborative filtering techniques in terms of scalability and performance. Web usage mining techniques such as clustering that rely on offline pattern discovery from user transactions can be used to improve the scalability of collaborative filtering, however, this is often at the cost of reduced recommendation accuracy. In this paper we propose effective and scalable techniques for Web personalization based on association rule discovery from usage data. Through detailed experimental evaluation on real usage data, we show that the proposed methodology can achieve better recommendation effectiveness, while maintaining a computational advantage over direct approaches to collaborative filtering such as the k-nearest-neighbor strategy.

1 Introduction

One of the most successful and widely used technologies for building personalization and recommendation systems is collaborative filtering (CF) [20]. Given a target user's record of activity, CF-based techniques, such as the k-Nearest-Neighbor (kNN) approach, compare that record with the historical records of other users in order to find the top k users who have similar tastes or interests. The mapping of a visitor record to its *neighborhood* could be based on similarity in ratings of items, access to similar pages, or purchase of similar items. The identified neighborhood is then used to recommend items not already accessed or purchased by the active user.

The CF-based techniques suffer from some well-known limitations [17]. For the most part these limitations are related to the scalability and efficiency of the kNN approach. Essentially, kNN requires that the neighborhood formation phase be performed as an online process, and for very large data sets this may lead to unacceptable latency for providing recommendations. A number of optimization strategies have been proposed and employed to remedy this shortcoming [3, 18]. These strategies include similarity indexing and dimensionality reduction to reduce real-time search costs.

The challenge in designing effective Web personalization systems is to improve the scalability of collaborative filtering through offline pattern discovery, while maintaining or improving the overall recommendation effectiveness. Furthermore, the effectiveness of the system must be measured in terms of both coverage and accuracy (precision) of the produced recommendations. Precision measures the degree to which the recommendation engine produces accurate recommendations. On the other hand, coverage measures the ability of the recommendation engine to produce all of the pageviews that are likely to be visited by the user. Both of these measures are essential in evaluating the effectiveness of recommender systems. For example, in e-commerce domain, low precision can easily lead to angry or frustrated users (who receive inaccurate recommendations) while low coverage will result in the site missing cross-sell or up-sell recommendations at critical junctures in users navigation through the site.

In recent years there has been an increasing interest and a growing body of work in Web usage mining [16] as an underlying approach to capturing and modeling Web user behavioral patterns and for deriving

e-business intelligence. Web usage mining techniques such as clustering that rely on offline pattern discovery from user transactions can be used to improve the scalability of collaborative filtering. For example, previous work such as [9, 10, 12] have considered automatic personalization based on clustering of user transactions and pageviews. However, this is often at the cost of reduced recommendation accuracy. One solution to improve accuracy is presented by [11] using preprocessing techniques such as normalization and significance filtering. Another way is to consider ordering information in personalization. Comparing with non-sequential patterns such as clusters and association rules, sequential patterns contain more precise information about user's navigational behavior. The use of navigational sequential patterns for predictive user modeling has been extensively studied [5, 14, 19]. The primary focus of all of these studies has been on prefetching of Web pages (i.e., predicting a user's next access to a page) to improve server performance or network latency. In the context of personalization, however, the narrow focus on navigational sequences often leads to very low recommendation coverage making such techniques less effective as the basis for recommender systems.

Some recent studies have considered the use of association rule mining [2, 15] in recommender systems [6, 7, 17]. For the most part, however, these studies have relied on discovering all association rules prior to generating recommendations (thus requiring search among all rules during the recommendation phase) or on real-time generation of association rules from a subset of transactions within a current user's neighborhood. There has also been little focus on the impact of factors such as the support threshold or the size of user history on the effectiveness of recommendations.

In this paper we present a scalable framework for recommender systems using association rule mining from clickstream data. Specifically, we present a data structure for storing the discovered frequent itemsets which is especially suitable for recommender systems. Our recommendation algorithm utilizes this data structure to produce recommendations efficiently in real-time, without the need to generate all association rules from frequent itemsets. Furthermore, through detailed experimental evaluation we show that by using multiple support levels for diferent types of pageviews and varying sized user histories, our framework can overcome some of the shortcomings of recommender systems based on association rules (e.g., low coverage resulting from high support thresholds or larger user histories, and reduced accuracy due to the sparse nature of the data). In fact, we show that the proposed framework can achieve better overall recommendation effectiveness than direct approaches such as the kNN technique in terms of coverage and accuracy.

2 A Framework for Personalization Based on Association Rules

Generally speaking, usage-based Web personalization systems [10] involve 3 phases: data preparation and transformation, pattern discovery, and recommendation. Of these, the latter is a real-time component, while the other two phases are performed offline. The pattern discovery phase may include the discovery of association rules, sequential navigational patterns, clustering of users or sessions, and clustering of pageviews or products. The recommendation engine considers the active user session in conjunction with the discovered patterns to provide personalized content. The personalized content can take the form of recommended links or products, or targeted advertisements tailored to the user's perceived preferences as determined by the matching usage patterns. In this paper, we focus on a specific instance of this general framework in which the recomendations are produced based on matching the current user session against patterns discovered through association rule mining on user transaction data. First, we briefly discuss the data preparation and pattern discovery phases and then we focus on the details of our recommendation engine.

2.1 Data Preparation and Pattern Discovery

The starting and critical point for successful personalization based on usage data is data preprocessing. The required high-level tasks are data cleaning, user identification, session identification, pageview identification, and the inference of missing references due to caching. Transaction identification can be performed as a final preprocessing step prior to pattern discovery in order to focus on the relevant subsets of pageviews in each user session. *Pageview* identification is the task of determining which page file accesses contribute to a single browser display. For Web sites using cookies or embedded session IDs, user and session identification is trivial. Web sites without the benefit of additional information for user and session identification must rely on heuristics methods. These heuristics and details of usage preprocessing tasks are explained in [4] and we do not discuss them further in this paper.

The above preprocessing tasks ultimately result in a set of n pageviews, $P = \{p_1, p_2, \cdots, p_n\}$, and a set of m user transactions, $T = \{t_1, t_2, \cdots, t_m\}$, where each $t_i \in T$ is a subset of P. Conceptually, we view each transaction t as an l-length sequence of ordered pairs:

$$t = \left\langle (p_1^t, w(p_1^t)), (p_2^t, w(p_2^t)), \cdots, (p_l^t, w(p_l^t)) \right\rangle,$$

where each $p_i^t = p_j$ for some $j \in \{1, \cdots, n\}$, and $w(p_i^t)$ is the weight associated with pageview p_i^t in the transaction t. The weights can be determined in a number of ways, however in the context of personalization based on clickstream data, the primary sources of data are server access logs. This allows us to choose two types of weights for pageviews: weights can be binary, representing the existence or non-existence of a product-purchase or a documents access in the transaction; or they can be a function of the duration of the associated pageview in the user's session. In this paper, since our focus is on association rule mining, we only consider binary weights on pageviews within user transactions, and furthermore, we ignore the ordering among the pageviews. Thus, a transaction can be viewed as a set of pageviews $s_t = \{p_i^t \mid 1 \leq i \leq l \text{ and } w(p_i^t) = 1\}$.

Association rules capture the relationships among items based on their patterns of co-occurrence across transactions. In the case of Web transactions, association rules capture relationships among pageviews based on the navigational patterns of users. For the current paper we have used the Apriori algorithm [2, 15] which follows a generate-and-test methodology. The Apriori algorithm, initially finds groups of items (which in this case are the pageviews appearing in the preprocessed log) occurring frequently together in many transactions (i.e., satisfying a user specified minimum support threshold). Such groups of items are referred to as *frequent item sets.*

Given a transaction set T and a set $I = \{I_1, I_2, \cdots, I_k\}$ of frequent itemsets over T, the support of an itemset $I_i \in I$ is defined as

$$\sigma(I_i) = \frac{|\{t \in T : I_i \subseteq t\}|}{|T|}$$

An important property of support is its downward closure: if an item set does not satisfy the minimum support criteria, then neither do any of its supersets. This property is essential for pruning the search space during each iteration of the Apriori algorithm.

Association rules which satisfy a minimum *confidence* threshold are then generated from the frequent itemsets. An association rule r is an expression of the form $X \Rightarrow Y(\sigma_r, \alpha_r)$, where X and Y are itemsets, σ_r is the support of $X \cup Y$, and α_r is the confidence for the rule r given by $\sigma(X \cup Y)/\sigma(X)$.

A problem with using a global minimum support threshold is that the discovered patterns will not include "rare" but important items which may not occur frequently in the transaction data. This is particularly important in the current context: when dealing with Web usage data, it is often the case that references to deeper content or product-oriented pages occur far less frequently that those of top level navigation-oriented pages. Yet, for effective Web personalization, it is important to capture patterns and generate recommendations that contain these items. Liu et al. [8] proposed a mining method with multiple minimum support that allows users to specify different support values for different items. In this method, the support of an itemset is defined as the minimum support of all items contained in the itemset. The specification of multiple minimum supports allows frequent itemsets to potentially contain rare items which are nevertheless deemed important. Our experimental results, presented in the next Section, show that the use of multiple support association rules can maintain the overall precision of recommendations, while dramatically improving coverage.

2.2 The Recommendation Engine

The recommendation engine takes a collection of frequent itemsets as input and generates a recommendation set for a user by matching the current user's activity against the discovered patterns. The recommendation engine is on-line process, therefore its efficiency and scalability are of paramount importance. In this section, we represent a data structure for storing frequent itemset and a recommendation generation algorithm which uses this data structure to directly produce real-time recommendations from itemsets without the need to first generate association rules.

We use a fixed-size sliding window over the current active session to capture the current user's history depth. For example, if the current session (with a window size of 3) is $< A, B, C >$, and the user references

T1: $\{ABDE\}$
T2: $\{ABECD\}$
T3: $\{ABEC\}$
T4: $\{BEBAC\}$
T5: $\{DABEC\}$

Table 1: Sample Web transactions involving pageviews A, B, C, D, and E

Size 1	Size 2	Size 3	Size 4
$\{A\}(5)$	$\{A,B\}(5)$	$\{A,B,C\}(4)$	$\{A,B,C,E\}(4)$
$\{B\}(6)$	$\{A,C\}(4)$	$\{A,B,E\}(5)$	
$\{C\}(4)$	$\{A,E\}(5)$	$\{A,C,E\}(4)$	
$\{E\}(5)$	$\{B,C\}(4)$	$\{B,C,E\}(4)$	
	$\{B,E\}(5)$		
	$\{C,E\}(4)$		

Table 2: Frequent Itemsets generated by the Apriori algorithm

the pageview D, then the new active session becomes $< B, C, D >$. This makes sense in the context of personalization since most users go back and forth while navigating a site to find the desired information, and it may not be appropriate to use earlier portions of the user's history to generate recommendations. Thus, the sliding window of size n over the active session allows only the last n visited pages to influence the recommendation value of items in the recommendation set.

The recommendation engine matches the current user session window with itemsets to find candidate pageviews for giving recommendations. Given an active session window w, we only consider all itemsets of size $|w| + 1$ satisfying a specified support threshold and containing the current session window. The recommendation value of each candidate pageview is based on the confidence of the corresponding association rule whose consequent is the singleton containing the pageview to be recommended. If the rule satisfies a specified confidence threshold requirement, then the candidate pageview is added to the recommendation set.

In order to facilitate the search for itemsets (of size $|w| + 1$) containing the current session window w, during the mining process the discovered itemsets are stored in a directed acyclic graph, here called a *Frequent Itemset Graph*. The Frequent Itemset Graph is an extension of the lexicographic tree used in the tree projection algorithm of [1]. The graph is organized into levels from 0 to k, where k is the maximum size among all frequent itemsets. Each node at depth d in the graph corresponds to an itemset, I, of size d and is linked to itemsets of size $d + 1$ that contain I at level $d + 1$. The single root node at level 0 corresponds to the empty itemset. To be able to match different orderings of an active session with frequent itemsets, all itemsets are sorted in lexicographic order before being inserted into the graph. The user's active session is also sorted in the same manner before matching with patterns.

Given an active user session window w, sorted in lexicographic order, a depth-first search of the Frequent Itemset Graph is performed to level $|w|$. If a match is found, then the children of the matching node n containing w are used to generate candidate recommendations. Each child node of n corresponds to a frequent itemset $w \cup \{p\}$. In each case, the pageview p is added to the recommendation set if the support ratio $\sigma(w \cup \{p\})/\sigma(w)$ is greater than or equal to α, where α is a minimum confidence threshold. Note that $\sigma(w \cup \{p\})/\sigma(w)$ is the confidence of the association rule $w \Rightarrow \{p\}$. The confidence of this rule is also used as the recommendation score for pageview p. It is easy to observe that in this algorithm the search process requires only $O(|w|)$ time given active session window w.

To illustrate the process, consider the example transaction set given in Table 1. Using these transactions, the Apriori algorithm with a frequency threshold of 4 (minimum support of 0.8) generates the itemsets given in Table 2. Figure 1 shows the Frequent Itemsets Graph constructed based on the frequent itemsets in Table 2. Now, given user active session window $< B, E >$, the recommendation generation algorithm finds items A and C as candidate recommendations. The recommendation scores of item A and C are 1 and 4/5,

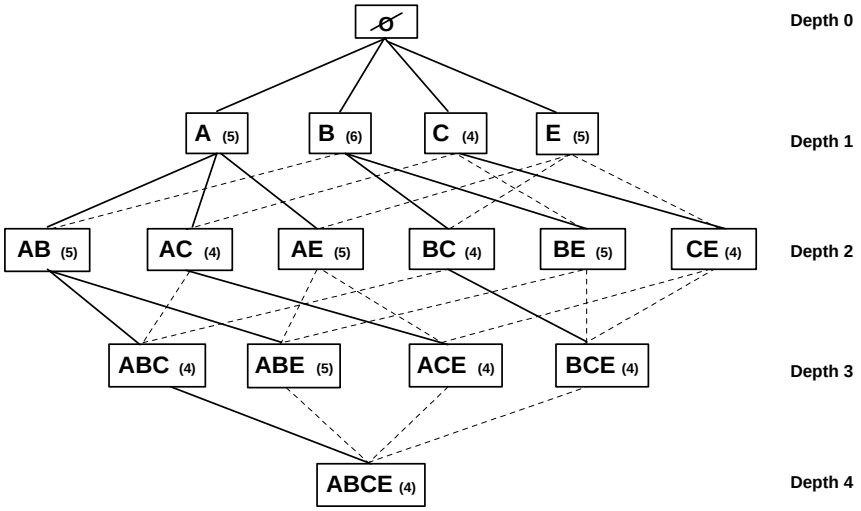

Figure 1: The Frequent Itemsets Graph for the example

corresponding to the confidences of the rules $\{B, E\} \rightarrow \{A\}$ and $\{B, E\} \rightarrow \{C\}$, respectively.

It should be noted that, depending on the specified support threshold, it might be difficult to find large enough itemsets that could be used for providing recommendations, leading to reduced coverage. This is particularly true for sites with very small average session sizes. An alternative to reducing the support threshold in such cases would be to reduce the session window size. This latter choice may itself lead to some undesired effects since we may not be taking enough of the user's activity history into account. Generally, in the context of recommendation systems, using a larger window size over the active session can achieve the better prediction accuracy. But, as in the case of higher support threshold, larger window sizes also lead to lower recommendation coverage. In order to overcome this problem, we use *all-kth-order* method proposed in [14] in the context of Markov chain models. In Markov models, the order of the model corresponds to the number of prior events used in predicting a future event. The use of all-kth-order Markov models generally requires the generation of separate models for each of the k orders, often leading to high space complexity.

Our algorithm is extended to generate all-kth-order recommendations as follows. First, the recommendation engine uses the largest possible active session window as an input for recommendation engine. If the engine cannot generate any recommendations, the size of active session window is iteratively decreased until a recommendation is generated or the window size becomes 0. We also note that, in contrast to standard all-kth-order Markov models, our framework does not require additional storage since all the necessary information (for all values of k) is captured by the Frequent Itemset Graph described above.

3 Experimental Evaluation

In this section, we discuss the experimental data set and present our evaluation metrics for recommendation effectiveness, and the results of our study based on these metrics.

3.1 Experimental Setup and Evaluation Metrics

For our experiments, we used the access logs from the Web site of the Association for Consumer Research (ACR) Newsletter (www.acr-news.org). After data preprocessing, the data set contains a total of 18342 transactions and 122 URLs. For our analysis, we eliminate both pageviews which appear in less than 0.5% or more than 80% of transactions and short transactions which contain less than 6 pageviews. This data set was divided into a training set and an evaluation set. After these preprocessing steps, the total number of remaining pageview URLs was 40. Approximately 70% of transactions were randomly selected as training set and the remaining transactions were used for evaluation.

Our evaluation methodology is as follows. Each transaction t in the evaluation set is divided into two parts. The first n pageviews in t are used for generating recommendations, whereas, the remaining portion of t is used to evaluate the generated recommendations. The value n reflects the maximum allowable window size for the experiments (in our case 4). Given a window size $w \leq n$, we select a subset of the first n pageviews as the surrogate for a user's *active session window*. The active session window is the portion of the user's clickstream used by the recommendation engine in order to produce a recommendation set. We call this portion of the transaction t the *active session with respect to* t, denoted by as_t. The recommendation engine takes as_t and a recommendation threshold τ as inputs and produce a set of pageviews as recommendations. We denote this recommendation set by $R(as_t, \tau)$. Note that $R(as_t, \tau)$ contains all pageviews whose recommendation score is at least τ (in particular, if $\tau = 0$, then $R(as_t, \tau) = P$, where P is the set of all pageviews).

The set of pageviews $R(as_t, \tau)$ can now be compared with the remaining $|t| - n$, pageviews in t. We denote this portion of t by $eval_t$. Our comparison of these sets is based on 2 different metrics, namely, precision and coverage. The *precision* of $R(as_t, \tau)$ is defined as:

$$precision(R(as_t, \tau)) = \frac{|R(as_t, \tau) \cap eval_t|}{|R(as_t, \tau)|},$$

and the *coverage* of $R(as_t, \tau)$ is defined as:

$$coverage(R(as_t, \tau)) = \frac{|R(as_t, \tau) \cap eval_t|}{|eval_t|}.$$

Precision measures the degree to which the recommendation engine produces accurate recommendations (i.e., the proportion of relevant recommendations to the total number of recommendations). Coverage measures the ability of the recommendation engine to produce all of the pageviews that are likely to be visited by the user (i.e., the proportion of relevant recommendations to all pageviews that should be recommended).

Finally, for a given recommendation threshold τ, the mean over all transactions in the evaluation set was computed as the overall evaluation score for each measure. We ran each set of experiments for thresholds ranging from 0.1 to 1.0. The results of these experiments are presented below.

3.2 Experimental Results

In all experiments we measured both precision and coverage of recommendations against varying recommendation thresholds from 0.1 to 1.0. To consider the impact of window size (the portion of user histories used to produce recommendations) we performed all experiments using window sizes of 1 through 4. Furthermore, we considered the impact of a global support threshold by varying the minimum support across all experiments. We compared these results with those produced by utilizing multiple support thresholds and all-kth-order recommendation model. Finally, we performed experiments to show the relative performance of our framework against the kNN technique for collaborative filtering. For the kNN method we chose $k = 20$ which seemed to provide the best overall results for the current data set, and we used the standard vector-space cosine similarity measure to generate nearest neighbors (from the training set) for the current active user session in the evaluation set.

Figure 2 shows the impact of window size on precision and coverage of recommendations. The results show clearly that precision increases as a larger portion of user's history is used to generate recommendations. Coverage, on the other hand is inversely affected by window size, although at higher recommendation thresholds the difference between various window sizes becomes smaller.

As expected, the experiments on the impact of support showed that a higher minimum support threshold during the mining stage results in lower coverage (but only slightly better precision). These results are not shown here. In general, it is desirable to use higher support thresholds in order to keep the model size small and to ensure the scalability of the association rule mining algorithm. However, as noted earlier, the higher support will result in missing some potentially important, yet infrequent, items as part of the recommendation set. In the context of Web personalization with clickstream data, the missed pageviews tend to be those that are particularly important (e.g., deeper content-oriented or product pages). Using the multiple support version of Apriori [8] helps alleviate this problem. Figure 3 shows the impact of using multiple support

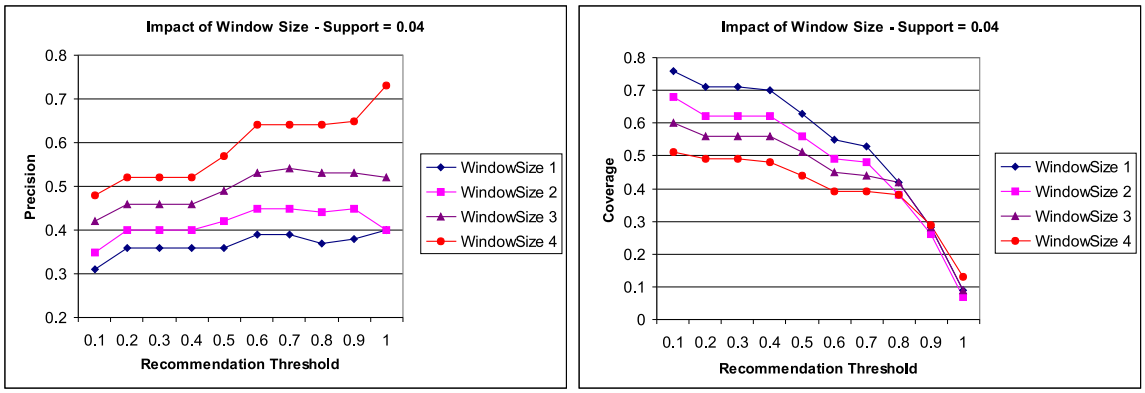

Figure 2: The Impact of Window Size on Coverage and Precision of Recommendations

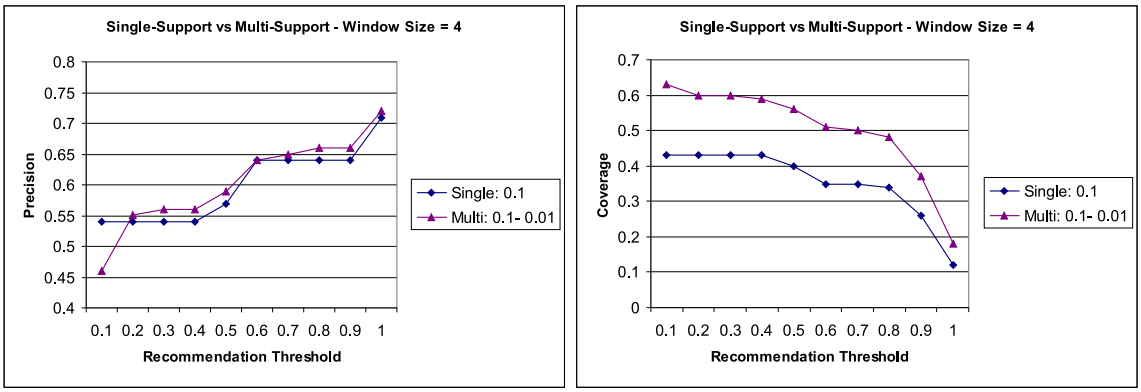

Figure 3: Comparison of Recommendation Effectiveness with Single Global Minimum Support Versus Multiple Support Thresholds

levels. In this experiment we selected several of the content-oriented pages situated more deeply in the site and assigned a minimum support of 0.01 to these pages. The other (navigational) pages were assigned a high support value of 0.1. The results are compared to using a single global minimum support threshold of 0.1. As the results suggest, the use of multiple support thresholds maintains the overall precision of recommendations while dramatically increasing the overall coverage (even at high recommendation thresholds).

The use of all-kth-order models (i.e., using varying-sized windows over the active session) has a similar impact as the use of multiple support levels. Figure 4 shows that all-kth-order model achieves similar precision (or better for higher recommendation thresholds) while improving the coverage of recommendations. The figure depicts the results for window size 3 and support threshold of 0.04, however, the relative results were similar for other combinations of window sizes and support values.

Figure 5 depicts the comparison of recommendation effectiveness between the kNN method for collaborative filtering and the combined association rule framework (including multiple support levels and all-kth-order recommendation model). The results for window size 4 show dramatic improvement in precision and an overall improvement in terms of coverage. The comparison with other window sizes (not shown) indicated that with increasing window size, the advantages of the association rule based method for both metrics become more pronounced.

4 Conclusions

Providing effective recommendations based on clickstream data can be important at early stages of a user's interaction (when more explicit user input such as ratings or personal perfiles are not available). Personal-

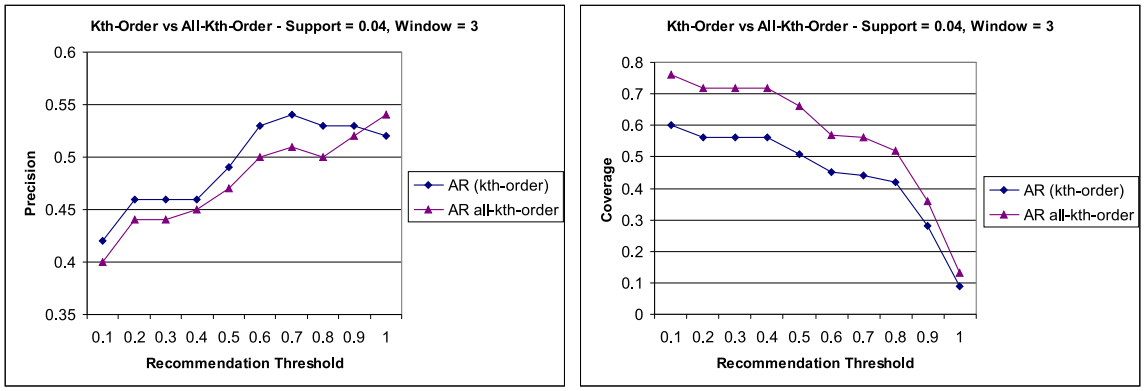

Figure 4: Comparison of Recommendation Effectiveness Using Fixed User Histories (kth-order) Versus Varying-Sized User Histories (all-kth-order)

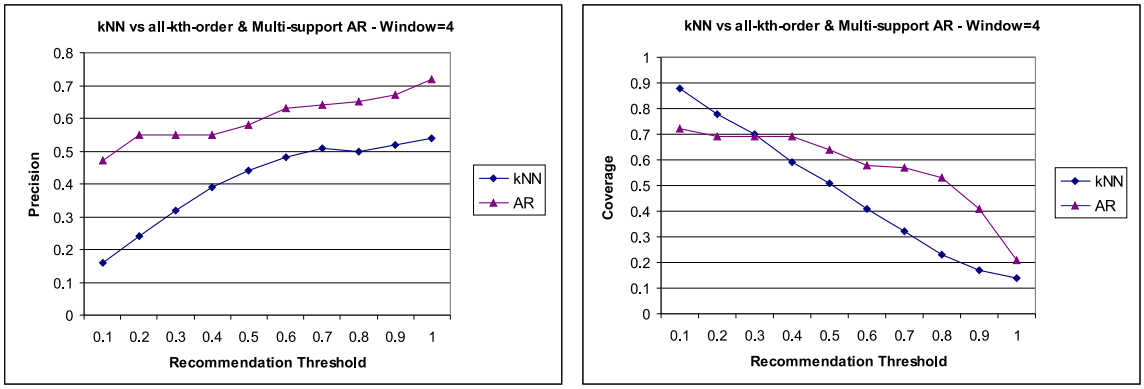

Figure 5: Relative Performance of Association Rule Recommendations Versus kNN Approach

ization at this level results in these user's engaging at a deeper level, and can help improve the conversion efficiency of a site. Standard collaborative filtering techniques such as kNN require real-time comparison of a current users record with the historical records of other users. This methodology becomes increasing unscalable as the number of users and items increase. The lack of scalability of kNN becomes amplified when dealing with the large volume of clickstream data.

In this paper we have presented a scalable framework for Web personalization based on association rule mining from clickstream data. Our framework includes an efficient data structure for storing frequent itemsets combined with a recommendation algorithm which allows for the generation of recommendations without first generating all association rules from itemsets. We have also studied the impact of using multiple support levels for different types of pageviews, as well as the use of varying-sized user histories on the precision and coverage of the generated recommendations. Our results show that the proposed framework can provide an effective alternative to standard collaborative filtering mechanism for personalization. In particular, we have shown that the association-based recommendation framework can, in fact, improve on the kNN-based collaborative filtering both in terms of the precision and coverage of recommendations, while at the same time maintaining the computational advantage over kNN attained due to the offline discovery of frequent patterns.

References

[1] R. Agarwal, C. Aggarwal, and V. Prasad. A tree projection algorithm for generation of frequent itemsets. *In Proceedings of the High Performance Data Mining Workshop*, Puerto Rico, 1999.

[2] R. Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conference on Very Large Data Bases, VLDB94*, 1994.

[3] C. C. Aggarwal, J.L. Wof, P. S. Yu. A new method for similarity indexing for market data. In *Proceedings of the ACM SIGMOD Conference*, 1999.

[4] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining World Wide Web browsing patterns. *Journal of Knowledge and Information Systems*, (1) 1, 1999.

[5] M. Deshpande and G. Karypis. Selective Markov models for predicting Web-page accesses. Technical Report #00-056, University of Minessota, 2000.

[6] X. Fu, J. Budzik, and K. J. Hammond. Mining navigation history for recommendation. In *Proc. 2000 International Conference on Intelligent User Interfaces*, New Orleans, January 2000. ACM.

[7] W. Lin, S.A. Alvarez, C. Ruiz. Collaborative recommendation via adaptive association rule mining. In *Proceedings of the Web Mining for E-Commerce Workshop (WebKDD'2000)*, August 2000, Boston.

[8] B. Liu, W. Hsu, and Y. Ma. Association rules with multiple minimum supports. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-99, poster)*, San Diego, CA, August 1999.

[9] B. Mobasher, R. Cooley, and J. Srivastava. Creating adaptive web sites through usage-based clustering of urls. In *IEEE Knowledge and Data Engineering Workshop (KDEX'99)*, November 1999.

[10] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on Web usage mining. In *Communications of the ACM*, (43) 8, August 2000.

[11] B. Mobasher, H. Dai, T. Luo and M. Nakagawa. Improving the effectiveness of collaborative filtering on anonymous Web usage data. In *Proceedings of the IJCAI 2001 Workshop on Intelligent Techniques for Web Personalization (ITWP01)*, August 2001, Seattle.

[12] B. Mobasher, H. Dai, T. Luo, M. Nakagawa, Y. Sun, and J. Wiltshire. Discovery of aggregate usage profiles for Web personalization. In *Proceedings of the WebKDD 2000 Workshop* at the ACM SIGKKD 2000, Boston, August 2000.

[13] M. Perkowitz and O. Etzioni. Adaptive Web sites: automatically synthesizing Web pages. In *Proceedings of Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.

[14] J. Pitkow and P. Pirolli. Mining longest repeating subsequences to Predict WWW Surfing. In *Proceedings of the 1999 USENIX Annual Technical Conference*, 1999.

[15] R. Srikant and R. Agrawal. Mining generalized association rules. In *Proceedings of the 21st Int'l Conference on Very Large Databases (VLDB95)*, Zurich, Switzerland, September 1995.

[16] J. Srivastava, R. Cooley, M. Deshpande, P-T. Tan. Web usage mining: discovery and applications of usage patterns from Web data. *SIGKDD Explorations*, (1) 2, 2000.

[17] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommender algorithms for e-commerce. In *Proceedings of the 2nd ACM E-Commerce Conference (EC'00)*, October 2000, Minneapolis.

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender systems: a case study. In *Proceedings of the WebKDD 2000 Workshop* at the ACM SIGKKD 2000, Boston, August 2000.

[19] S. Schechter, M. Krishnan, and M. D. Smith. Using path profiles to predict http requests. *In Proc. 7th International World Wide Web Conference*, April 1998, Brisbane, Australia.

[20] U. Shardanand, P. Maes. Social information filtering: algorithms for automating "word of mouth." In *Proceedings of the ACM CHI Conference (CHI95)*, 1995.