Improving the Effectiveness of Collaborative Filtering on Anonymous 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

Recommender systems based on collaborative filtering usually require real-time comparison of users' ratings on objects. In the context of Web personalization, particularly at the early stages of a visitor's interaction with the site (i.e., before registration or authentication), recommender systems must rely on anonymous clickstream data. The lack of explicit user ratings and the shear amount of data in such a setting poses serious challenges to standard collaborative filtering techniques in terms of scalability and performance. Offline clustering of users 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 study the impact of various preprocessing techniques applied to clickstream data, such as clustering, normalization, and significance filtering, on collaborative filtering. Our experimental results, performed on real usage data, indicate that with proper data preparation, the clustering-based approach to collaborative filtering can achieve dramatic improvements in terms of recommendation effectiveness, while maintaining the computational advantage over the direct approaches such as the k-Nearest-Neighbor technique.

1 Introduction

In today's highly competitive e-commerce environment, the success of a site often depends on the site's ability to retain visitors and turn casual browsers into potential customers. Automatic personalization and recommender system technologies have become critical tools in this arena since they help tailor the site's interaction with a visitor to his or her needs and interests. Current technologies for personalization rely on explicit or implicit expressions of user interests (such as product clickthroughs or purchases, product ratings, or profile information obtained through a registration process). Obtaining such information about users, usually requires fairly deep interactions of these users with the site, even for first-time visitors.

To engage visitors at a very early stage (for example before registration and identification), personalization tools can only rely on the visitors' clickstream data and on very short user histories. Yet, this type of *anonymous personalization* is desirable since personalized interactions with a visitor at this stage makes it more likely that the visitor will stay and engage at a deeper level.

One of the most successful and widely used technologies for building personalization and recommendation systems is collaborative filtering (CF) [4; 17]. 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 content or pages, or purchase of similar items. The identified neighborhood is then used to recommend items not already accessed or purchased by the active user. Thus, there are two primary phases in collaborative filtering: the *neighborhood formation* phase and the *recommendation* phase.

The CF-based techniques suffer from some well-known limitations [13]. 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 [1; 11; 14; 18; 19]. These strategies include similarity indexing and dimensionality reduction to reduce real-time search costs, as well as offline clustering of user records to reduce the online component of the system to search within a matching cluster.

In the context of personalization using anonymous clickstream data, our goal is to provide meaningful recommendations to users at the earliest stage possible in their interactions with the site. In this setting, collaborative filtering faces some additional challenges. These challenges emanate from the characteristics of Web usage data, including the following:

- available feature values are not item ratings, rather they are either binary (indicating a visit or a non-visit for a particular pageview) or they are a function of the time spent on a particular pageview;
- while the user-item matrix in this setting is generally more dense than in the case of user ratings, predictions still have to be based on very short user histories (trails), sometime involving no more than 2-3 clickthroughs;
- in case of binary feature values, a one (i.e., a visit to a page) is not necessarily an indication of interest in that item;
- the sequential nature of the clickstream makes the order in which pages are visited potentially (but, not necessarily) relevant in making predictions; and
- the number of user records (and items) can be far larger than the standard domains for CF where users are limited to purchasers or people who have rated items (thus, making optimization even a more important criteria).

Our objective in this paper is to assess the effectiveness of collaborative filtering applied to anonymous usage data. To improve the scalability of CF we first perform clustering on user sessions to form candidate neighborhoods. This clustering is performed offline and independent of any targeted user. The recommendation engine will then compare a portion of an active user's session to representatives for the discovered clusters in order to obtain recommendations. We call these cluster representatives *aggregate usage profiles.*

Our expectation is that, while improving scalability, CF in conjuction with aggregate usage profiles would lead to a drop in the accuracy of predictions when compared to the direct approach (i.e., kNN without clustering). Thus, our primary focus will be on various data preparation procedures, such as normalization and significance filtering, which can potentially improve the effectiveness of the clustering approach to collaborative filtering in the context of anonymous clickstream data. Our experimental results indicate that with proper preprocessing, CF based on usage profiles, can achieve effectiveness in par with the direct approach, while dramatically improving the scalability. The practical significance of these results is that they show how effective, yet anonymous, Web personalization can be achieved based solely on users' clickstream data, particularly at early stages in their visits.

2 Personalization Based on Anonymous Web Usage Data

In recent years there has been an increasing interest in personalization based on Web usage mining as a way to remedy shortcomings of existing approaches [6]. These shortcomings include reliance on subjective user profiles which may be prone to biases, or on standing profiles which may become outdated with changing user needs or interests. In particular, Web usage mining has been suggested as an enabling mechanism for improving and optimizing the structure of a site [12; 15], improving the scalability and performance of Web-based recommender systems [6; 9], and discovering better e-business intelligence for the purpose of online marketing [2]. For an up-to-date survey of Web usage mining techniques and systems see [16].

Generally speaking, usage-based Web personalization systems 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. A general framework for usage-based Web personalization was presented in [6]. In [8], this framework was extended to incorporate text and content mining for more effective and useful recommendations. 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 the case of kNN-based collaborative filtering, there is no offline pattern discovery component. Instead, the active user session is compared directly with all other user sessions previously recorded in order to form its neighborhood. The recommendation set can then be determined based on a weighted average of feature values for items not already visited by the user. As noted earlier, often in practice, the offline data preparation component of the system may involve performing similarity indexing or dimensionality reduction, in order to speed up the online neighborhood formation and recommendation phases in kNN.

2.1 Data Preparation for Clickstream Personalization

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. The difficulties involved in identifying users and sessions depend greatly on the server-side technologies used for the Web site. 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. We use the heuristics proposed in [3] to identify unique user sessions from anonymous usage data and to infer cached references.

Pageview identification is the task of determining which page file accesses contribute to a single browser display, and is heavily dependent on the intra-page structure, and hence requires detailed site structure information. The significance of a pageview may depend on usage, content and structural characteristics of the site, as well as prior domain knowledge specified by the site designer. For example, in an in an e-commerce site pageviews corresponding to product-oriented events (e.g., shopping cart changes) may be considered more significant than others.

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 anonymous personalization, 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 the case of time durations, it should be noted that usually the time spent by a user on the last page visited in the session is not available. We set the weight for the last page to be the mean time duration for the page taken across all sessions in which the page does not occur as the last pageview.

For the clustering tasks as well as for collaborative filtering based on the kNN technique, we represent each user transaction as a vector over the n-dimensional space of pageviews, where feature values are the weights of these pageviews in the corresponding transaction. Thus given the transaction t above, the l-length transaction vector $\overrightarrow{t}$ is given by:

$$\overrightarrow{t} = \left\langle w_{p_1}^t, w_{p_2}^t, \cdots, w_{p_n}^t \right\rangle,$$

where each $w_{p_j}^t = w(p_i^t)$, for some $i \in \{1, \cdots, n\}$, in case p_j appears in the transaction t, and $w_{p_j}^t = 0$, otherwise.

Given this representation, the set of all user transactions can be viewed as an $m \times n$ transaction-pageview matrix.

Significance Testing

Using binary weights in the representation of user transactions is often desirable due to efficiency requirements in terms of storage and computation of similarity coefficients among transactions. However, in this context, it becomes more important to determine the significance of each pageview or item access. For example, a user may access an item p only to find that he/she is not interested in that item, subsequently backtracking to another section of the site. We would like to capture this behavior by discounting the access to p as an insignificant access. In general, it is not sufficient to filter out pageviews with small durations since the amount of time spent by users on a page is not only based on the user's interest on the page. The page duration may also be dependent on the characteristics and the content of the page. For example, we would expect that users spend far less time on navigational pages than they do on content or product-oriented pages.

Statistical significance testing can help capture some of the semantics illustrated above. Basically, the weight associated with an item in a transaction should be 0, if the amount of time spent on that item is significantly below the mean time duration of the item across all user transactions. More formally, for each transaction vector $\overrightarrow{t} = \left\langle w_{p_1}^t, w_{p_2}^t, \cdots, w_{p_n}^t \right\rangle$, where weights are time durations, we perform the following transformation:

$$sig(\overrightarrow{t}) = \left\langle v_{p_i}^t \mid 1 \leq i \leq n \right\rangle,$$

where,

$$v_{p_i}^t = \begin{cases} 1, & \text{if } zscore(w_{p_i}^t) \geq \theta \\ 0, & \text{otherwise} \end{cases}$$

and

$$zscore(w_{p_i}^t) = \frac{w_{p_i}^t - mean(p_i)}{\sigma(p_i)}.$$

The threshold θ is the significance threshold, and $mean(p_i)$ and $\sigma(p_i)$ are the mean time duration and the standard deviation for pageview p_i across all transactions, respectively.

The resulting binary transaction-pageview matrix can then be used to perform similarity computations for clustering and kNN neighborhood formation tasks.

Normalization

There are also some advantages in using the fully weighted representation of transaction vectors (based on time durations). One advantage is that for many distance- or similarity-based clustering algorithms, more granularity in feature weights usually leads to more accurate results. Another advantage is that, since relative time durations are taken into account, the need for performing other types of preprocessing, such as significance testing, is greatly reduced.

However, as indicated by our experiments, raw time durations may not be an appropriate measure for the significance of a pageview. This is because a variety factors, such as structure, length, and the type of pageview, as well as the user's interests in a particular item, may affect the amount of time spent on that item. Appropriate weight normalization can play an essential role in correcting for these factors.

We consider two types of weight normalization applied to user transaction vectors: normalization across pageviews in a single transaction and normalization of a pageview weight across all transactions. We call these transformations *transaction normalization* and *pageview normalization*, respectively. Specifically, for transaction normalization, we apply the following transformation to each transaction vector $\vec{t} = \langle w^t_{p_1}, w^t_{p_2}, \cdots, w^t_{p_n} \rangle$:

$$norm_t(\vec{t}) = \langle v^t_{p_i} \mid 1 \leq i \leq n \rangle ,$$

where,

$$v^t_{p_i} = \frac{w^t_{p_i} - \min\limits_{1\leq j\leq n} \left\{ w^t_{p_j} \right\}}{\max\limits_{1\leq j\leq n} \left\{ w^t_{p_j} \right\} - \min\limits_{1\leq j\leq n} \left\{ w^t_{p_j} \right\}}.$$

On the other hand, pageview normalization is performed as follows:

$$norm_p(\vec{t}) = \langle v^t_{p_i} \mid 1 \leq i \leq n \rangle ,$$

where,

$$v^t_{p_i} = \frac{w^t_{p_i} - \min\limits_{t\in T} \left\{ w^t_{p_i} \right\}}{\max\limits_{t\in T} \left\{ w^t_{p_i} \right\} - \min\limits_{t\in T} \left\{ w^t_{p_i} \right\}}.$$

Pageview normalization is useful in capturing the relative weight of a pageview for a user with respect to the weights of the same pageview for all other users. On the other hand, transaction normalization captures the importance of a pageview to a particular user relative to the other items visited by that user in the same transaction. The latter is particularly useful in focusing on the "target" pages in the context of short user histories. In our experimental results we examine the impact of the transformations described above on kNN as well as on collaborative filtering based on aggregate usage profiles.

2.2 Collaborative Filtering with Clickstream Data

Just as in the case of user transactions, a current user's active session can be viewed as an n-dimensional vector over the space of pageviews. In this case, also, the weights associated with pageviews within the active session can be binary or based on time durations. Thus an active session vector is represented as $\vec{s} = \langle w^s_{p_1}, w^s_{p_2}, \cdots, w^s_{p_n} \rangle$. The types of transformations discussed in the previous section for user transactions can also be applied to this active session vector.

For a particular user a, the active session $\vec{s_a}$ can be incrementally updated as the user visits additional items. One important consideration in Web personalization based on clickstream data is which portion of the user's trail should be used to provide recommendations. In general, it may not be appropriate to use the whole clickstream trail for this purpose because most users navigate several paths leading to independent pieces of information within a session. In many cases these "subsessions" have a length of no more than 2 or 3 references. We capture the user history depth within a sliding window over the current session. The sliding window of size r over the active session allows only the last r visited pages to influence the recommendation value of items in the recommendation set.

The kNN-Based Approach

In the case of kNN, we measure the similarity or correlation between the active session $\vec{s}$ and each transaction vector $\vec{t}$ (where $t \in T$). The top k most similar transactions to $\vec{s}$ are considered to be the neighborhood for the session s, which we denote by $NB(s)$ (taking the size k of the neighborhood to be implicit):

$$NB(s) = \{ \vec{t}_{s_1}, \vec{t}_{s_2}, \cdots, \vec{t}_{s_k} \}.$$

A variety of similarity measures can be used to find the nearest neighbors. In traditional collaborative filtering domains (where feature weights are item ratings on a discrete scale), the Pearson r correlation coefficient is commonly used. This measure is based on the deviations of users' ratings on various items from their mean ratings on all rated items. However, this measure is not appropriate in the context of anonymous personalization based on clickstream data (particularly in the case of binary weights). Instead we use the *cosine* coefficient, commonly used in information retrieval, which measures the cosine of the angle between two vectors. The cosine coefficient can be computed by normalizing the dot product of two vectors with respect to their vector norms. Given the active session $\vec{s}$ and a transaction $\vec{t}$, the similarity between them is obtained by:

$$sim(\vec{t}, \vec{s}) = \frac{\vec{t} \cdot \vec{s}}{|\vec{t}| \times |\vec{s}|}.$$

In order to determine the which items (not already visited by the user in the active session) are to be recommended, a recommendation score is computed for each pageview $p_i \in P$ based on the neighborhood for the active session. Two factors are used in determining this recommendation score: the overall similarity of the active session to the neighborhood as a whole, and the average weight of each item in the neighborhood.

First we compute the mean vector (centroid) of $NB(s)$. The feature value for each pageview in the mean vector is computed by finding the ratio of the sum of the pageview's weights across transactions to the total number of transactions in the neighborhood. We denote this vector by $cent(NB(s))$. For each pageview p in the neighborhood centroid, we can now obtain a recommendation score as a function of the similarity of the active session to the centroid vector and the weight of that item in this centroid. In our implementation, we have chosen to use the following function, denoted by $rec(\vec{s}, p)$:

$$rec(\vec{s}, p) = \sqrt{weight(p, NB(s)) \times sim(\vec{s}, cent(NB(s)))},$$

where $weight(p, NB(s))$ is the mean weight for pageview p in the neighborhood as expressed in the centroid vector. If the pageview p is in the current active session, then its recommendation value is set to zero.

If a fixed number N of recommendations are desired, then the top N items with the highest recommendation scores are considered to be part of the recommendation set. In our implementation, we normalize the recommendation scores for all pageviews in the neighborhood (so that the maximum recommendation score is 1), and return only those which satisfy a threshold test. In this way, we can compare the performance of kNN across different recommendation thresholds.

The Clustering-Based Approach
Given the mapping of user transactions into a multi-dimensional space as vectors of pageviews, standard clustering algorithms, such as k-means, generally partition this space into groups of transactions that are close to each other based on a measure of distance or similarity. Such a clustering will result in a set $TC = \{c_1, c_2, \cdots, c_k\}$ of transaction clusters, where each c_i is a subset of the set of transactions T. Ideally, each cluster represents a group of users with similar navigational patterns. However, transaction clusters by themselves are not an effective means of capturing an aggregated view of common user patterns. Each transaction cluster may potentially contain thousands of user transactions involving hundreds of pageview references. Our ultimate goal in clustering user transactions is to reduce these clusters into weighted collections of pageviews. We call these collections *aggregate usage profiles*.

In [9], we introduced and evaluated a technique for the derivation of aggregate usage profiles from transaction clusters. We call this method PACT (Profile Aggregation Based on Clustering Transactions). This is the approach we use in the current set of experiments. In the simplest case, PACT generates aggregate profiles based on the centroids of each transaction cluster, in much the same way as described above for the neighborhoods of an active session in the kNN approach.

In general, PACT can consider a number of other factors in determining the item weights within each profile, and in determining the recommendation scores. These additional factors may include the link distance of pageviews to the current user location within the site or the rank of the profile in terms of its significance. However, to be able to consistently compare the performance of the clustering-based approach to that of kNN, we restrict the item weights to be the mean feature values of the transaction cluster centroids. In this context, the only difference between PACT and the kNN-based approach is that we discover transaction clusters offline and independent of a particular target user session.

To summarize the PACT method, given a transaction cluster c, we construct a usage profile pr_c as a set of pageview-weight pairs:

$$pr_c = \{\langle p, weight(p, pr_c)\rangle \mid p \in P, weight(p, pr_c) \geq \mu\},$$

where the significance weight, $weight(p, pr_c)$, of the pageview p within the usage profile pr_c is:

$$weight(p, pr_c) = \frac{1}{|c|} \cdot \sum_{t \in c} w_p^t,$$

and w_p^t is the weight of pageview p in transaction $t \in c$. The threshold parameter μ is used to prune out very low support pageviews in the profile.

This process results in a number of aggregate profiles each of which can, in turn, be represented as a vector in the original n-dimensional space of pageviews. The real-time component of the system can compute the similarity of an active session $\vec{s}$ with each of the discovered aggregate profiles. The top matching profile is used to produce a recommendation set in a manner similar to that for the kNN approach discussed above. If $\vec{pr}$ is the vector representation of the top matching profile pr, we compute the recommendation score for the pageview p by

$$rec(\vec{s}, p) = \sqrt{weight(p, pr) \times sim(\vec{s}, \vec{pr})},$$

where $weight(p, pr)$ is the weight for pageview p in the profile pr. As in the case of kNN, if the pageview p is in the current active session, then its recommendation value is set to zero.

Clearly, collaborative filtering based on PACT will result in dramatic improvement in scalability and computational performance, since most of the computational cost is incurred during the offline clustering phase. We would expect, however, that this decrease in computational costs be accompanied also by a decrease in recommendation effectiveness. Our goal is to improve the effectiveness of PACT, through the preprocessing transformations discussed above, while maintaining the computational advantages.

3 Experimental Evaluation

For our evaluation we used the access logs from the Web site of the Association for Consumer Research (ACR) Newsletter (www.acr-news.org). In this section we measure and compare the performance to kNN and PACT both with and without the application various transformations such as significance testing and normalization.

3.1 Experimental Setup

We selected a portion of the ACR log files, which after preprocessing and removing references by Web spiders, resulted in approximately 12,000 sessions. Support filtering was used to eliminate pageviews appearing in less than 0.5% or more than 80% of transactions (including the site entry page). Furthermore, for these experiments we eliminated short transactions, leaving only transactions with at least 6 pageviews. Approximately 1/3 of these transactions were randomly selected as the evaluation set, and the remaining portion was used as the training set for profile generation and neighborhood formation in PACT and kNN, respectively. The total number of remaining pageview URLs in the training and the evaluation sets was 40. For the PACT method, we used multivariate k-means clustering to partition the transaction file. Overlapping aggregate profiles were generated from transaction clusters using the method described earlier.

Our evaluation methodology is as follows. For each transaction t in the evaluation set, we select the first n pageviews in t 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. Both of the CF-based techniques take 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 portion of t, i.e., with $t - as_t$, to measure the recommendation effectiveness using 3 different metrics, namely, precision, coverage, and the F1 measure. The *precision* of $R(as_t, \tau)$ is defined as:

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

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

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

precision measures the degree to which the recommendation engine produces accurate recommendations (i.e., recommends pageviews that will have be visited by the user in the remaining portion of the user's session). 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. Ideally, one would like high precision and high coverage. A single measure that captures this is the F1 measure [5], defined in terms of precision and coverage:

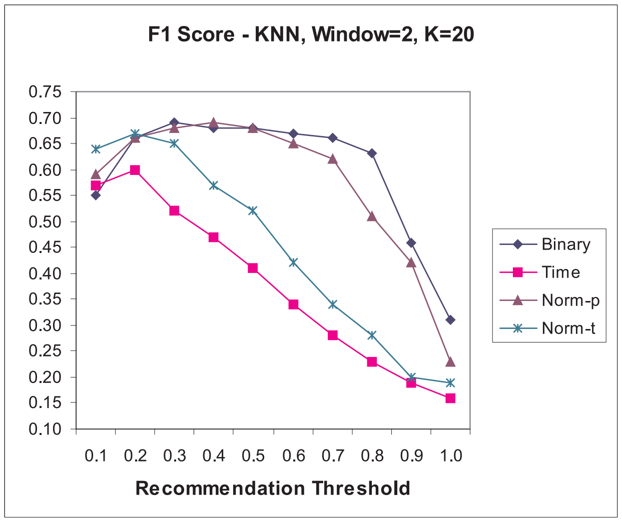
$$F1 = \frac{2 \times precision \times coverage}{precision + coverage}.$$

The F1 measure attains its maximum value when both precision and coverage are maximized.

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

Since our focus is on collaborative filtering with clickstream data based on very short user trails, in all of the experiments we set the active session window size to 2. This means that only the first two pageviews of each evaluation transaction was used to generate recommendations. For the kNN method we chose $k = 20$ which seemed to provide the best overall results for the current data set. Figure 1 depicts the performance of kNN based on the precision and the F1 measures (we do not show the coverage results as they can be inferred by observation from the other two measures).



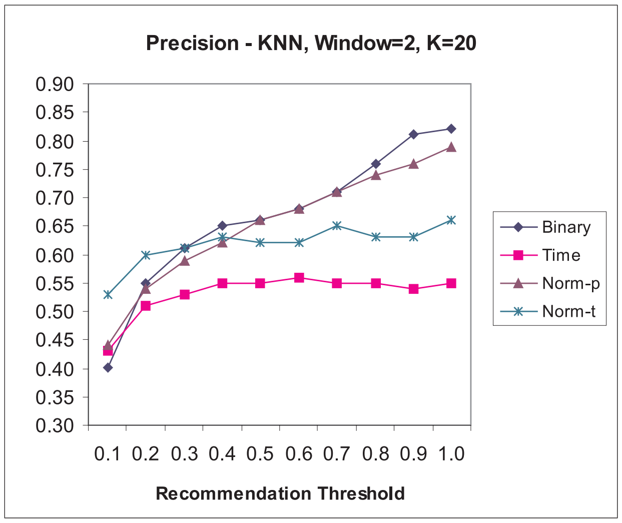
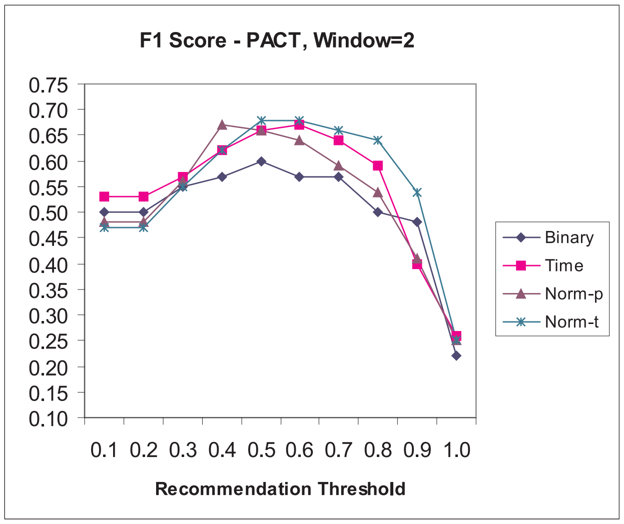


Figure 1: Performance of kNN on clickstream data, using binary weights, raw time weights, and normalized time weights.

Surprisingly, kNN achieved the best results with simple binary weights. Its performance was rather poor, however, when raw time weights were used for feature values. This can be attributed to the fact that, within a certain range, there is great deal of variation in the amount of time users spend on pages even when they consider these pages significant. With raw term weights, some users who should be considered close neighbors could potentially be missed. In this respect, we expect the clustering-based algorithms to perform better due to a higher level of aggregation captured by clusters. The results also indicate that weight normalization resulted in marked improvement. Particularly, in the case of pageview normalization, kNN achieved performance in par with the binary case.



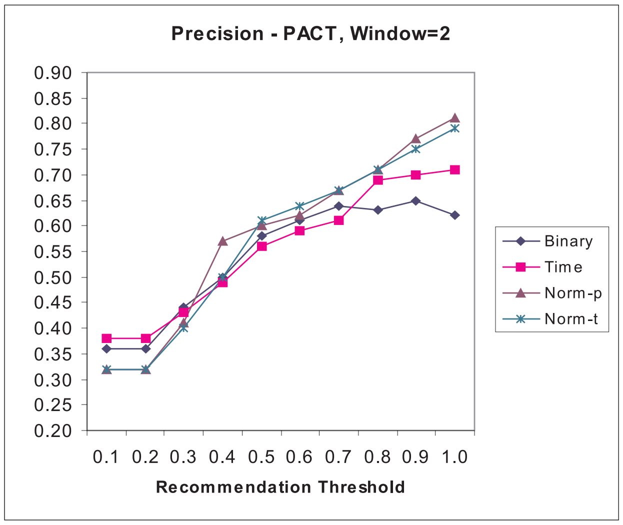
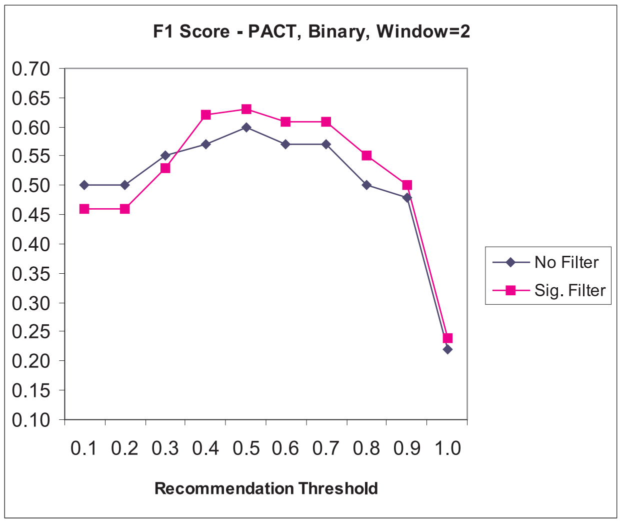


Figure 2: Performance of PACT on clickstream data, using binary weights, raw time weights, and normalized time weights.



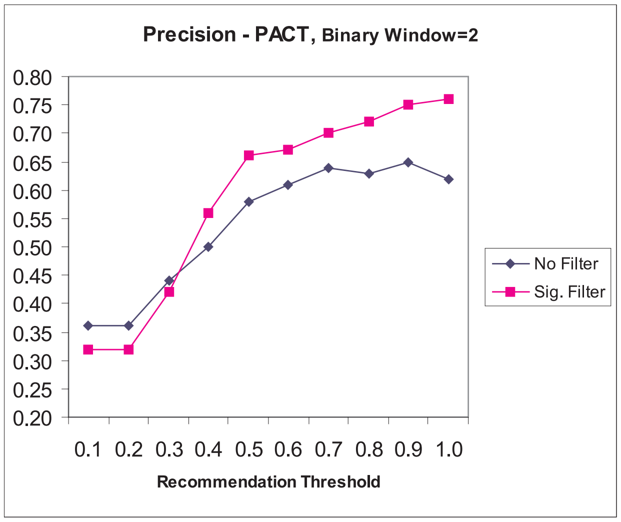


Figure 3: Performance improvement of PACT on clickstream data when significance filtering is performed on the transaction data.

Figure 2 shows the normalization results for PACT. In the binary case, PACT did worse than kNN in terms of precision and coverage, while, as expected it performed better using raw term weights. The most significant observation here is that normalization dramatically improved recommendation effectiveness, especially at higher thresholds. In fact, with normalization, PACT achieved similar performance as the best case in kNN. Transaction and page normalization both gave similar results. In general, however, the specific type of normalization is highly dependent on the data distribution.

For the evaluation of significance filtering we chose the value of the z-score factor to be 2.4 which provided the best results. In other words, prior to conversion to the binary representation, we filtered out pageviews from transactions, if the time duration for those pageviews was more than 2.4 standard deviations lower than the mean. Figure 3 depicts these results. Just as the case of normalization, significance filtering provides dramatic improvements for PACT when compared to the binary base case. The significance filtering results for kNN also showed improvement, however, the best results were still achieved with the binary data. We do not show these results here.

Overall, these (and other) experiments show that transformations such as pageview or transaction normalization and significance filtering can have a great impact on the effectiveness of clustering-based CF techniques, and can be used to regain the loss of accuracy suffered by these techniques relative to more direct approaches such as kNN.

4 Conclusions

Many of the existing real-time Web personalization and recommender systems are based on collaborative filtering. Traditional collaborative filtering techniques, such as, kNN suffer from limitations emanating from their lack of scalability and effectiveness in the face of very large and sparse data sets. Offline clustering of user transactions can dramatically improve the efficiency of such systems. However, this is often at the cost of decreased accuracy. In the case of anonymous Web usage data, there is also the challenge of accurately predicting user interests based on very short user clickstream trails, and without the benefit of more intimate user information.

In this paper, we have explored several data preparation techniques, namely, normalization of pageview weights in user transactions and significance filtering, and studied their impact on the effectiveness of collaborative filtering on anonymous clickstream data. We have provided detailed evaluation results, based on real usage data, which show that with proper data transformation at the preprocessing stage we can significantly improve the effectiveness of collaborative filtering based on clustering while retaining the computational advantage this approach provides over the kNN-based techniques.

References

[1] 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.

[2] A. Buchner and M. D. Mulvenna. Discovering internet marketing intelligence through online analytical Web usage mining. *SIGMOD Record*, (4) 27, 1999.

[3] 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.

[4] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval*, August 1999.

[5] D. Lewis, W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual ACM-SIGIR Conference*, (3) 12, London, UK, Springer-Verlag, 1994.

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

[7] 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.

[8] B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Zhu. Integrating Web usage and content mining for more effective personalization, in *E-Commerce and Web Technologies,*" *Lecture Notes in Computer Science (LNCS)* 1875, Springer-Verlag, September 2000.

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

[10] B. Mobasher. A Web personalization engine based on user transaction clustering. In *Proceedings of the 9th Workshop on Information Technologies and Systems (WITS'99)*, December 1999.

[11] M. O'Conner, J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems*, Berkeley, CA, 1999.

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

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the ACM Conference on E-Commerce (EC00)*, Minneapolis, October 2000.

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

[15] M. Spiliopoulou, C. Pohle, and L. C. Faulstich. Improving the effectiveness of a Web site with Web usage mining. In Workshop on Web Usage Analysis and User Profiling (WebKKD99), San Diego, August 1999.

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

[18] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *Proceedings of Workshop on Recommendation Systems at the 15th National Conference on Artificial Intelligence*, 1998.

[19] P. S. Yu. Data mining and personalization technologies. In *Proceedings of the Int'l Conference on Database Systems for Advanced Applications (DASFAA99)*, April 1999, Hsinchu, Taiwan.