Effective Next-Items Recommendation via Personalized Sequential Pattern Mining

Ghim-Eng Yap[1], Xiao-Li Li[1], and Philip S. Yu[2]

[1] Institute for Infocomm Research, 1 Fusionopolis Way #21-01 Connexis Singapore 138632
{geyap,xlli}@i2r.a-star.edu.sg
[2] Department of Computer Science, University of Illinois at Chicago, IL 60607-7053
psyu@cs.uic.edu

Abstract. Based on the intuition that frequent patterns can be used to predict the next few items that users would want to access, sequential pattern mining-based next-items recommendation algorithms have performed well in empirical studies including online product recommendation. However, most current methods do not perform *personalized* sequential pattern mining, and this seriously limits their capability to recommend the best next-items to each specific target user. In this paper, we introduce a personalized sequential pattern mining-based recommendation framework. Using a novel Competence Score measure, the proposed framework effectively learns user-specific sequence importance knowledge, and exploits this additional knowledge for accurate personalized recommendation. Experimental results on real-world datasets demonstrate that the proposed framework effectively improves the efficiency for mining sequential patterns, increases the user-relevance of the identified frequent patterns, and most importantly, generates significantly more accurate next-items recommendation for the target users.

1 Introduction

With the rapid growth of online information sources and e-commerce businesses, users increasingly need reliable recommender systems [25,4] to highlight relevant next-items, i.e., the next few items that the users most probably would like. Fortunately, the user consumption histories offer crucial clues that help us to tackle this important problem. Whenever we visit web pages or purchase things from online stores, we leave a time-ordered sequence of items that we have seen or bought. When these historical sequences are consolidated into a *sequence database* (SDB), we can employ a powerful data mining process - *sequential pattern mining* (SPM) [1] - to discover temporal patterns that are frequently repeated among different users. Sequential pattern mining-based next-items recommendation works on the intuition that if many users have accessed an item j *after* an item i, it makes sense to recommend item j to someone who just seen item i.

Most sequential pattern mining algorithms focus on efficiently finding *all* the sequential patterns with frequencies above a given threshold in a sequence database. For example, algorithms exist for exact sequential pattern mining [1,3,11,16,20,30,33], approximate mining [14], constraint-based mining [5,7,21], as well as sequential pattern mining from incremental [8] and progressive databases [13]. A significant shortcoming is that all current methods do not perform *user-specific* sequential pattern mining and, as a result, they cannot give accurate *personalized* recommendation to users.

S.-g. Lee et al. (Eds.): DASFAA 2012, Part II, LNCS 7239, pp. 48–64, 2012.
© Springer-Verlag Berlin Heidelberg 2012

In this paper, we address the important problem of *personalized* sequential pattern mining-based next-items recommendation, where the system discovers the *user-specific* frequent sequential patterns from past users' sequences, and it uses the mined patterns to predict the next few items which a target user would access. We establish the theoretical relation between (1) *support* of a sequential pattern p which recommends an item i, and (2) *predictive power* of p in terms of whether i is visited. Our analysis confirms that the predictive power of high-support (*frequent*) patterns is bounded by greater information gain [9], which justifies the *inductive assumption* [18] in sequential pattern-based recommendation - that the higher-support sequential patterns predict next-items better.

Current sequential pattern-based recommendation methods are ineffective because the simple frequency counts of patterns are used to compute the *support* they get from the sequence database (SDB). These methods treat all sequences in the SDB as equally important. However, in real applications, sequences in SDB often carry varying significance (or *weights*) with respect to each target user. To mine the *personalized* sequential patterns for a target user, we have to effectively model this varying relevance among the historical sequences for that specific user. Since each sequence in the SDB belongs to a different user, we can weight the sequences based on available knowledge about, for example, the target user's social affiliation to other users, or in most cases, how relevant is each of the sequences compared to the known sequence of the target user. For instance, in a book recommender system, the transaction records (i.e., sequences) of users who have borrowed similar books as the target user should be assigned greater weights because they are more likely to lead to accurate recommendations. Likewise, web browsing sequences with pages related to those the target user has visited should carry greater weights when we mine for sequential patterns to make recommendation.

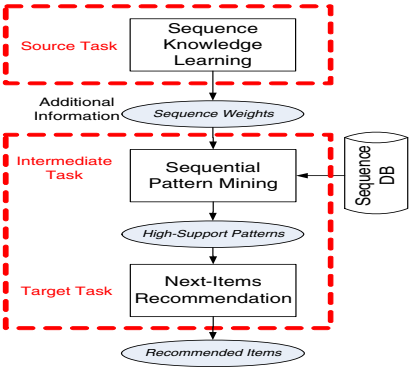


Fig. 1. Proposed personalized learning framework - effective learning is achieved by providing sequential pattern mining with an additional information source (the personalized sequence weights from a related source task) apart from the standard training data (the sequence database (DB))

We shall show that significantly more accurate next-items recommendations can be achieved through using our novel framework (Figure 1) to enable *personalized sequence weight learning and sequential pattern mining-based next-items recommendation*.

The proposed framework personalizes the sequence database (SDB) by assigning a user-specific weight to each sequence in the SDB (*source task*). This effectively enables the sequential pattern mining algorithms to perform a personalized search in sequence space for user-specific frequent patterns (*intermediate task*), which leads to significantly more accurate personalized next-items recommendation (*target task*). This approach is modeled after the inherent ability of human to recognize and leverage additional knowledge from related *source tasks* to improve the performance of a *target task* [31]. The next-items recommendation performance improves due to the sequence weight learning that makes the resulting sequential patterns much more relevant to every target user.

Our framework is novel because it *inverts the current paradigm by advocating the deep mining of user-specific patterns*, unlike traditional methods which generate all frequent patterns and prune them according to user relevance. The traditional post-filtering methods will not work well for *personalized* recommendation because the meaningful patterns for one user may not be frequent among the sequences of all users, so the traditional methods miss important patterns that really match the target user's preferences. By exploiting sequence weight knowledge to adjust the support of patterns, we effectively personalize the hypothesis space in which frequent patterns are searched, thereby enabling the discovery of more user-relevant frequent sequential patterns which cannot be found by traditional sequential pattern mining algorithms. The resulting user-specific sequential patterns are then more useful for personalized next-items recommendation.

We summarize our major research contributions as follows:

1. We propose a novel personalized sequential pattern mining-based next-items recommendation framework that learns and exploits additional user-specific sequence importance knowledge to improve the accuracy of next-items recommendation.
2. We propose a novel Competence Score to learn user-specific sequence weights for mining personalized frequent patterns and to recommend user-relevant next-items.
3. Through experimental validation using real-world sequence datasets, we demonstrate that learning of user-specific sequence weights is scalable, and it yields significantly more accurate personalized next-items recommendation for the target users.

The rest of this paper is organized as follows. Section 2 discusses the related work and Section 3 formally defines the *personalized sequential pattern mining-based next-items recommendation* problem. Section 4 introduces our novel personalized recommendation framework, and Section 5 presents extensive empirical results on real-world datasets to validate the proposed framework. Finally, Section 6 concludes this paper.

2 Related Works

The sequential pattern mining-based next-items recommendation is an inductive learning task where the objective is to induce a predictive model (i.e., the next-items recommender system) from a set of examples (i.e., the sequences in database). The usefulness of sequential pattern mining-based recommendation has, to a certain extent, been

Table 1. An example sequence database

No.	Sequence
1	<(1,4),(3),(2,8),(1,5)>
2	<(5,6),(1,2),(4,9),(3),(8)>
3	<(5),(7),(1,6),(3),(2),(8)>

demonstrated empirically by past studies on various domains such as e-commerce [12], web browsing [34] and IPTV programs scheduling [23], but there remains a lack of theoretical analysis on why high-support sequential patterns are useful for recommending next-items to users. We provide the necessary theoretical justification in this paper.

The proposed framework leverages on the learned user-specific sequence weights to improve the next-items recommendation. This is similar in concept to Bayesian transfer, where inductive transfer improves Bayesian learning [31]. In Bayesian learning, the *prior* distribution is combined with training data to get the predictive model. Bayesian transfer provides informative priors as additional knowledge from source task, so that the resulting model is more accurate. The learning of sequence weights for sequential pattern mining-based recommendation is similar to the learning of priors for Bayesian prediction in that the additional sequence weight knowledge helps the sequential pattern mining algorithms to better interpret the data and thus yield better recommendations.

Weighted sequential pattern mining [17,32] recognizes the difference in importance between sequences, but assumes that the importance of a sequence depends only on its items. The importance of items are derived from their domain value (e.g. price and popularity). The same weights are assigned to sequences and patterns with the same item characteristics regardless of the target user, which obviously cannot generate accurate *personalized predictions* since user-specific information are not used. Capelle et al. [5] proposed a method where they require the discovered patterns to be similar to a reference pattern. Specifically, they mandate that each user has to supply a reference pattern as well as specify a minimum similarity threshold. The similarity constraint introduces restrictions that only their proposed sequential pattern mining algorithm can satisfy, making their solution non-generalizable to other sequential pattern mining algorithms. In contrast, our framework extends to any source task that is related to sequence weight learning (not restricted to reference sequence matching and no need to ask users to state the similarity threshold etc), and can be readily integrated with all the existing sequential pattern mining algorithms. Additionally, our novel Competence Score method not only emphasizes other users' sequences that are relevant to the target user by computing their compatibility scores, but also takes the sequences' recommendation ability (or extensibility) into consideration, thus giving much better next-items recommendations.

3 Problem Definition

We now formally introduce the personalized sequential pattern mining-based next-items recommendation problem. A sequence database (SDB) is a collection of sequences, i.e. SDB=$\{s_1, s_2, \ldots, s_n\}$ where $|SDB| = n$ denotes the number of sequences (also the number of users as each user has a corresponding sequence in SDB). A sequence $s_i \in$

$SDB(i = 1, 2, \ldots, n)$, can be represented as $s_i = <s_{i1}, s_{i2}, \ldots, s_{i|s_i|}>$, which is an ordered list of itemsets (each s_{ij} in s_i $(j = 1, 2, \ldots, |s_i|)$ is an itemset) associated with an user U_i. Each itemset s_{ij} comprises transacted items with the same timestamp $t(s_{ij})$, and the different itemsets s_{ij1} and s_{ij2} in the same sequence s_i cannot have the same timestamp, i.e., $t(s_{ij1}) \neq t(s_{ij2}), (j_1 \neq j_2)$. A sequence $s_i = <s_{i1}, s_{i2}, \ldots, s_{i|s_i|}>$ contains a sequence $s_j = <s_{j1}, s_{j2}, \ldots, s_{j|s_j|}>$, or s_j is a *subsequence* of s_i, if there exist integers $d_1 < d_2 < \ldots < d_{|s_j|}$ such that $s_{j1} \subseteq s_{id1}, s_{j2} \subseteq s_{id2}, \ldots, s_{j|s_j|} \subseteq s_{id|s_j|}$ [1]. Table 1 shows an example sequence database comprising three sequences. Each row is a sequence representing a user's consumption histories; items in each ()-bracket form an itemset with same timestamp, e.g. (1,4). Sequence 1 describes a user who accessed the items 1 and 4 at timestamp t_0, item 3 at timestamp t_1, items 2 and 8 at timestamp t_2, and items 1 and 5 at timestamp t_3, etc. Using *patterns* to refer to subsequences generated during mining process, the *count* of a pattern p is the number of sequences in SDB containing p. The *support* for pattern p is then defined as $support(p) = count(p)/|SDB|$.

Given a target user's past sequence s_q and the database of all other users' past sequences, the next-items recommendation task is to predict items that the target user is most likely to access *in the near future*, i.e., those items in the next few itemsets that he or she will access. We do not restrict the prediction to just the immediate next item that a target user will access, nor the immediate itemset that the user will access in the next visit [24]. The *personalized* next-items recommendation task can be formalized by identifying and ranking the candidate next-items through a novel framework of personalized sequential pattern mining that takes into account the relative importance (i.e., weights) of other users' sequences in the database according to each target user, so that the top-m items (may be from multiple itemsets) are recommended to the target user.

4 Personalized Sequential Pattern Mining-Based Recommendation

Given a sequence database SDB and a minimum support δ, the traditional sequential pattern mining task is to discover all the patterns having a support not less than δ, under the assumption that all the sequences in the SDB should be treated equally. The required most-probable next-items are then recommended from the patterns having the highest-support. In contrast, *the problem of next-items recommendation via personalized sequence weight learning takes into account the fact that sequences in the real-world have different importance to each specific user; sequence weights should be effectively learned and exploited to generate more user-relevant sequential patterns, which can in turn significantly improve the next-items recommendation accuracy for the end-users*.

While numerous empirical studies (e.g. [12,34,23]) have demonstrated that sequential pattern mining-based next-items recommendation is effective in many different domains, we take the analysis further by formally explaining how the predictive power and the support of sequential patterns are related, similar to how the discriminative power of features in pattern-based classification is related to the pattern/feature frequency [9]. Specifically in our case, we need to establish that reliably computing the support values of patterns can result in better next-items recommendations, in order to explain why our proposed framework, which enables the resulting pattern supports to reliably reflect the user-relevance, is so effective in practice for personalized next-items recommendations.

4.1 The Predictive Power and Support of Patterns

Let $V \in \{0, 1\}$ and $X \in \{0, 1\}$ be the events of *visiting an item* i and *predicting item* i, respectively. The information gain (a standard measure of predictive power [9]) for V (i.e., whether i is visited) from knowing X (i.e., whether i is predicted) is computed by:

$$IG(V|X) = H(V) - H(V|X) \tag{1}$$

where $H(V)$ and $H(V|X)$ are the entropy and conditional entropy for V. Specifically,

$$H(V|X) = -\sum_X P(X) \sum_V P(V|X) log(P(V|X)) \tag{2}$$

Given a sequence database, $H(V)$ is fixed as the prior probability of i, so the upper bound of the information gain (i.e., $IG_{UB}(V|X)$) depends only on the lower bound of the conditional entropy for V given X (i.e., on $H_{LB}(V|X)$). $P(X = 1)$ is the probability of predicting item i. Item i is predicted if a pattern that predicts i is considered *frequent*, i.e., if $P(X = 1) = \theta \geq$ *minsup*, where *minsup* is a minimum support threshold and θ is the support of such a pattern. $P(V = 1)$ is the probability that i is visited. Given a sequence database, $P(V = 1) = \alpha$ is a fixed value equal to i's support in the database ($\alpha \geq 0$). By Apriori lemma, i appears in any pattern that predicts it, so $\theta \leq \alpha$.

Letting $P(V = 1|X = 1) = \beta$, the conditional entropy $H(V|X)$ is at its minimum if knowing $X = 1$ completely predicts V, i.e., if $\beta = 1$ or $\beta = 0$. For the case $\beta = 1$,

$$H_{LB}(V|X) = (\alpha - 1) log(\frac{1-\alpha}{1-\theta}) - (\alpha - \theta) log(\frac{\alpha-\theta}{1-\theta}) \tag{3}$$

The partial derivative of $H_{LB}(V|X)_{\beta=1}$ with respect to the support θ is:

$$\frac{\partial H_{LB}(V|X)_{\beta=1}}{\partial \theta} = log(\frac{\alpha-\theta}{1-\theta}) \leq log1 \leq 0 \tag{4}$$

So, $H_{LB}(V|X)$ reduces as the support θ increases. For the case $\beta = 0$,

$$H_{LB}(V|X) = (\theta - (1-\alpha)) log(\frac{1-\alpha-\theta}{1-\theta}) - \alpha log(\frac{\alpha}{1-\theta}) \tag{5}$$

The partial derivative of $H_{LB}(V|X)_{\beta=0}$ with respect to the support θ is:

$$\frac{\partial H_{LB}(V|X)_{\beta=0}}{\partial \theta} = log(\frac{1-\alpha-\theta}{1-\theta}) \leq log1 \leq 0 \tag{6}$$

The above analysis reveals that lower bound conditional entropy $H_{LB}(V|X)$ monotonically decreases as θ increases, so theoretical upper bound information gain increases with θ. Hence, the predictive power of a higher-support pattern is upper-bounded by a higher information gain, justifying the strategy of predicting items from more frequent sequential patterns rather than less frequent patterns. This explains why our proposed framework can be so effectively used for personalized next-items recommendation.

4.2 Sequence Weight Learning

We shall now discuss the source task which performs the sequence weight learning (Figure 1). Given a target user's sequence s_q, for each s_i in sequence database SDB ($s_i = <s_{i1}, s_{i2}, \ldots, s_{i|s_i|}>$ is user U_i's sequence), we assign a sequence weight $w(s_i, s_q)$ to s_i that represents its significance with respect to the target user sequence s_q. We first discuss two existing methods and then present our novel Competence Score method.

Method 1: User-independent Weight Learning. Weighted sequential pattern mining [6] weighs s_i based on the characteristics of items in s_i, e.g., price [32], popularity [23] (application oriented). These methods give the same weight for s_i regardless of the target user's sequence s_q. Weight $w(s_i, s_q)$ can be computed using the mean popularity of s_i's itemsets — popularity of an itemset s_{i_j}, $s_{i_j} \in s_i$, is computed as $\mathit{count}(s_{i_j})/|SDB|$.

Method 2: User-dependent Weight Learning. This method accounts for the differences between target users to learn more user-specific sequence weights. In this learning task, each $s_i \in SDB$ is compared to s_q to determine their similarity, and similar sequences in SDB are deemed more relevant for making recommendation [15,27] to the target user. The weight of s_i w.r.t. s_q is computed using the following pattern similarity functions:

1. Longest Common Subsequence (LCS) [29,5]: Let the longest common subsequence of s_i and s_q be denoted $LCS_{i,q} = <lcs_{i,q_1}, lcs_{i,q_2}, \ldots, lcs_{i,q_{|LCS_{i,q}|}}>$ (each lcs_{i,q_j} is an itemset that is common to sequences s_i and s_q, such that timestamp $t(lcs_{i,q_{j+1}}) > t(lcs_{i,q_j}), j = 1, 2, \ldots, |LCS_{i,q}| - 1$). The weight $w(s_i, s_q)$ is defined as:

$$w(s_i, s_q) = \frac{|LCS_{i,q}|}{|s_q|} \tag{7}$$

   where $|\cdot|$ denotes the number of itemsets in a sequence.
2. Cosine Similarity [23]: The weight $w(s_i, s_q)$ for s_i w.r.t. s_q can be measured as the cosine similarity [26] between their respective itemset vectors (denoted v_i and v_q):

$$w(s_i, s_q) = \frac{v_i \cdot v_q}{|v_i||v_q|} \tag{8}$$

   where v_i and v_q are the vector representations of s_i and s_q, respectively, such that each dimension in v_i and v_q denotes an unique itemset.

Method 3: Our Proposed Weight Learning Method. Our novel *Competence Score* method computes a personalized score for each sequence s_i in the sequence database (SDB) to reflect its competency in recommending next-items for the target user. Every s_i is a temporally-ordered list of itemsets with timestamps from oldest to most recent. Instinctively, s_i is highly-competent if it is not only compatible to the user sequence s_q, but also is capable of readily extending beyond s_q to offer more next-items. To satisfy both these requirements, our proposed Competence Score is computed in three steps:

1. **Step 1: Compute backward-compatibility score (BCS) of each sequence $s_i \in SDB$ w.r.t. the user sequence s_q.** We iterate over every sequence s_i in sequence database (SDB) to evaluate its *string-compatibility* and *temporal-compatibility* to s_q. String compatibility of s_i and s_q is derived by Longest Common Subsequence ($LCS_{i,q}$, Eq. 7), so that the longer the $LCS_{i,q}$, the more compatible s_i is to s_q. However, string-compatibility alone does not take into account that the most recently transacted itemset by the target user is the most relevant for predicting next-items to that user [34,17]. Our BCS score also considers the *temporal*-compatibility of the itemsets in s_i and s_q by weighting the itemsets in s_q such that its more recent itemsets are weighted heavier. This is analogous to a Markov Chain where the future items of the target user depend mainly on the most recent items the user saw.

   Letting the longest common subsequence of s_i and s_q be $LCS_{i,q} = < lcs_{i,q_1}, lcs_{i,q_2}, \ldots, lcs_{i,q_{|LCS_{i,q}|}} >$, the weight of each itemset $lcs_{i,q_j} \in LCS_{i,q}$ is $w(t_j, t_l)$, where t_j is the timestamp of lcs_{i,q_j} in s_q and t_l is the timestamp of the last itemset in s_q. For a given s_q, t_l is a constant so we can simply refer to the weight as $w(t_j)$. This itemset weight decay is modeled using the normalized Gaussian distribution:

$$w(t_j) = ae^{-\frac{(t_j - t_l)^2}{2\sigma^2}}, 0 \leq w(t_j) \leq 1 \tag{9}$$

   where $a = \frac{1}{\sigma\sqrt{2\pi}}$ and $\sigma = (t_f - t_l)/(2\sqrt{2\ln 2})$; the constants t_f and t_l denote the timestamps of the first and the last itemsets in the user sequence s_q, respectively.

   The time-weighted BCS of s_i w.r.t. target sequence s_q is computed as follows:

$$b_{i,q}(t) = \frac{\sum_{j=1}^{|LCS_{i,q}|} w(t_j), lcs_{i,q_j} \in LCS_{i,q}}{\sum w(t_k), s_{q_k} \in s_q}, 0 \leq b_{i,q}(t) \leq 1 \tag{10}$$

   where t_k is the timestamp of itemset s_{q_k} in s_q. This BCS score favors s_i sequences whose $LCS_{i,q}$ contains itemsets closer in time to the most recent itemset in s_q.
2. **Step 2: Compute the forward-extensibility score (FES) of each sequence $s_i \in SDB$ w.r.t. the user sequence s_q.** We define forward-extensibility score (FES) of s_i to answer the following important question: having seen some of the itemsets in s_q, does s_i have sufficient suitable next-items to recommend to the target user?

   Similar to our BCS score, our FES score computation involves the identification of the longest common subsequence ($LCS_{i,q}$) between each $s_i \in SDB$ and s_q. Let $LCS_{i,q} =< lcs_{i,q_1}, lcs_{i,q_2}, \ldots, lcs_{i,q_{|LCS_{i,q}|}} >$. We identify the set of unique candidate next-items that s_i is able to recommend to the target user, by finding the unique items from those itemsets in s_i that are later than the last common itemset $lcs_{i,q_{|LCS_{i,q}|}}$. The basic formula for computing the FES score is therefore:

$$f_{i,q} = \frac{|CNI_{i,q}|}{|CNI_{i,q}|_{max}}, 0 \leq f_{i,q} \leq 1 \tag{11}$$

   where $CNI_{i,q} = \{cni_{i,q_1}, cni_{i,q_2}, \ldots, cni_{i,q_{|CNI_{i,q}|}}\}$ is the set of unique candidate next-items in s_i with respect to s_q. To keep $f_{i,q}$ between 0 to 1, we only consider up to at most the first $|CNI_{i,q}|_{max}$ unique candidate next-items after $lcs_{i,q_{|LCS_{i,q}|}}$,

where $|CNI_{i,q}|_{max}$ is the maximum desired number of new items, a standard parameter in recommenders. In this way, $|CNI_{i,q}|$ (i.e., the number of candidate next-items in $CNI_{i,q}$) cannot be more than $|CNI_{i,q}|_{max}$. We further default parameter $|CNI_{i,q}|_{max}$ to the number of items in s_q, $||s_q||$, since it is reasonable for heavier users to demand more recommendations (e.g., books). The FES formula becomes:

$$f_{i,q} = \frac{|CNI_{i,q}|}{||s_q||}, 0 \leq f_{i,q} \leq 1 \tag{12}$$

where $||s_q||$ denotes the number of items in sequence s_q.

Furthermore, we need to take into account the different suitability of each candidate next-item cni_{i,q_j} ($cni_{i,q_j} \in CNI_{i,q}$) in terms of how much time difference exists between the timestamp t_k when the last common itemset $lcs_{i,q_{|LCS_{i,q}|}}$ (i.e., the last itemset in $LCS_{i,q}$) was seen in s_i and the timestamp t_j when candidate next-item cni_{i,q_j} was seen in s_i. The intuition is that items transacted nearer in time from the most recent common itemset (i.e., $lcs_{i,q_{|LCS_{i,q}|}}$) are more relevant to the target user and should get more weight, whilst candidate next-items that are further in time from $lcs_{i,q_{|LCS_{i,q}|}}$ are not quite as relevant to the target user and should be weighted less. This decay (over time) in item weight $w(t_j, t_k)$, which we simply refer to as $w(t_j)$ since t_k (timestamp of $lcs_{i,q_{|LCS_{i,q}|}}$ in s_i) is a constant given the sequence pair s_i and s_q, is modeled using the normalized Gaussian distribution:

$$w(t_j) = ae^{-\frac{(t_j - t_k)^2}{2\sigma^2}}, 0 \leq w(t_j) \leq 1 \tag{13}$$

where $a = \frac{1}{\sigma\sqrt{2\pi}}$ and $\sigma = (t_l - t_k)/(2\sqrt{2\ln 2})$; the constants t_l and t_k denote the timestamps of the last itemset in s_i and the last itemset in $LCS_{i,q}$, respectively.

The time-weighted FES of s_i w.r.t. target sequence s_q is computed as follows:

$$f_{i,q}(t) = \frac{\sum w(t_j), cni_{i,q_j} \in CNI_{i,q}}{||s_q||}, 0 \leq f_{i,q}(t) \leq 1 \tag{14}$$

where $||s_q||$ denotes number of items in sequence s_q.

3. **Step 3: Compute Competence Score (CS) of sequence s_i with respect to user sequence s_q by merging its BCS and FES**. The sequence weight $w(s_i, s_q)$ is given by the *Competence Score* ($c_{i,q}(t)$) of s_i w.r.t. s_q. We define s_i's Competence Score $c_{i,q}(t)$ as the harmonic mean between its time-sensitive BCS and FES scores:

$$c_{i,q}(t) = \frac{b_{i,q}(t) \times f_{i,q}(t)}{\frac{1}{2} \times (b_{i,q}(t) + f_{i,q}(t))}, 0 \leq c_{i,q}(t) \leq 1 \tag{15}$$

The BCS $b_{i,q}$ is high only if the given sequence s_i is highly-compatible to the target user's sequence, while the FES $f_{i,q}$ is high only if s_i has sufficient next-items to recommend. Being their harmonic mean, CS $c_{i,q}$ is high only if both $b_{i,q}$ and $f_{i,q}$ are high, i.e., both compatibility and extensibility requirements are satisfied. In specific domains where there exists a certain preferred trade-off between BCS and FES, the more general weighted harmonic mean (where different weights can be allocated to

BCS and FES) can also be used. Besides the harmonic mean, alternative methods to merge BCS and FES have been investigated but are found to be not as suitable. For instance, using multi-objective optimization (i.e., skyline queries [22]) may not maintain the balance between the two important requirements, e.g. a high skyline score may be given to a sequence s_i with a high BCS but a very low FES [19].

4.3 Exploiting the Sequence Weight Knowledge for Next-Items Recommendation

In the proposed framework, personalized weights are first assigned to all sequences in the sequence database (SDB) as an off-line operation using the aforementioned learning methods. We now define the personalized *count* and *support* to take into account the learned sequence weights so that the user-specific sequence knowledge is effectively exploited by the proposed framework to personalize the sequential pattern mining:

[*Pattern Count*]. Given that each sequence $s_i \in SDB$ has a learned weight $w(s_i, s_q)$ with respect to user sequence s_q, the count of pattern x in sequence database (SDB) is:

$$count(x, s_q) = \sum_{s_i:(x \subseteq s_i)} w(s_i, s_q) \tag{16}$$

Equation 16 sums the weights of all sequences which contain the pattern x, such that a higher $count(x, s_q)$ means a higher support for pattern x in the sequence database.

[*Pattern Support*]. The support of x in SDB is defined as:

$$support(x, s_q) = \frac{count(x, s_q)}{\sum_{s_i \in SDB} w(s_i, s_q)} \tag{17}$$

where the denominator is the total sequence weight in the sequence database.

The above support definition satisfies the monotonically-decreasing property: given patterns A and B, if $B \subseteq A$, then $support(B) \geq support(A)$, as B must be part of all sequences containing A. Hence, it is readily applicable to any of the conventional Apriori-inspired sequential pattern mining algorithms to mine the high-support (*frequent*) patterns (the intermediate task in Figure 1). An example of such sequential pattern mining algorithms is PrefixSpan [20], the popular pattern-growth approach in which a sequence database is recursively projected into a set of smaller databases, and sequential patterns are grown in each projected database by exploring only locally frequent fragments [20].

During sequential pattern mining, patterns with personalized *support* not less than the specified minimum support δ are output as the user-specific frequent patterns F_q:

$$F_q = \{x | support(x, s_q) \geq \delta, \delta > 0\} \tag{18}$$

In Eq. 18, we used sequence weight knowledge to eliminate all the user-irrelevant records in SDB which are not related to the target user (since we will not include x in F_q if $support(x, s_q) = 0$). As such, we can significantly improve the mining efficiency.

The frequent pattern set F_q is then used in our target task (Figure 1) to recommend next-items to the target user [34,17], where the most recent items of the target user are considered more valuable for predicting the next-items. Let the target user's known

sequence be $s_q = <i_1, i_2, \ldots, i_n>$ where i_j ($i_j \in s_q$) is an itemset. We find candidate pattern set $CP = \{x | i_n \in x, x \in F_q\}$ which is the frequent patterns containing i_n. We eliminate redundant patterns from CP by keeping only closed patterns starting with i_n.

Let a candidate pattern be $cp = <i_n, i_{n+1}, \ldots, i_{n+k}>$, $cp \in CP$. Items in the itemsets after i_n are candidate items for recommendation. The support for each candidate item i_c ($i_c \in i_w, w > n, i_w \in cp$) is the sum of *support* of all cp that promote i_c, i.e.,

$$support_{i_c} = \sum_{i_c \in i_w, w > n, i_w \in cp, cp \in CP} support(cp, s_q) \tag{19}$$

The overall algorithm is presented in Algorithm 1, where our proposed Competence Score method is used to learn the personalized sequence weights for each target user. The personalized sequence weights are then effectively exploited to discover the user-specific frequent sequential patterns for personalized next-items recommendation.

Algorithm 1. Personalized Sequential Pattern Mining-based Recommendation

INPUT: The target user u_q's sequence, s_q, and the database SDB of other users' sequences
BEGIN
`// sequence weight learning step (source task)`
for all sequence $s_i \in SDB$ **do**
Compute $b_{i,q}(t)$ // backward-compatibility (Eq. 10)
$f_{i,q}(t)$ // forward-extensibility (Eq. 14)
$c_{i,q}(t)$ // Competence Score (Eq. 15)
end for // now we have the SDB with sequence weights personalized to user u_q
`// sequential pattern mining step (intermediate task)`
Apply a state-of-the-art sequential pattern mining algorithm to get personalized frequent patterns, F_q, with Eqs. 16 and 17 to compute the personalized support
`// next-items recommendation step (target task)`
Use Eq. 19 to compute the personalized support of each candidate next-item in frequent patterns F_q, and recommend the items with highest support to user u_q
END

5 Experimental Evaluation

We present a two-part evaluation of our proposed personalized sequential pattern mining-based next-items recommendation framework (Figure 1). In the first part (Section 5.2), we evaluate the effectiveness of learning sequence weights with respect to the sequential pattern mining. In the second part (Section 5.3), we evaluate the framework's effectiveness in terms of prediction accuracy of the next-items recommendation.

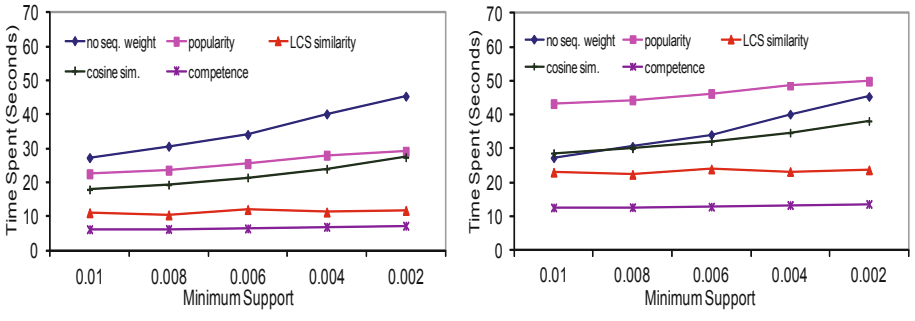
5.1 Experimental Setup

– **Algorithms:** We compare the sequence weight learning methods with the traditional sequential pattern mining-based recommendation [34,17] with no weight learning. The state-of-art PrefixSpan [20] algorithm is used for the pattern mining.

– **Datasets:** We use two real-world datasets for the performance evaluation. The first dataset is the msnbc.com dataset from UCI [2]. It captures the time-ordered sequence of webpages visited by msnbc users on a day. There are a total of 989,818 sequences, each one corresponding to a different user. The second dataset is a book-loan dataset comprising the borrowing records for 126,714 users on 144,966 books (126,714 sequences), all during a six-month period.
– **Evaluation Metrics:** The measures to evaluate if sequence weight learning helps the next-items recommendation include [31]: (1) reduction in time for mining sequential patterns, and (2) improvement in accuracy for next-items recommendation.

5.2 Evaluating the Framework Efficacy

We choose msnbc.com dataset to evaluate the efficacy of our framework as it has much more sequences (close to 1 million) than the book-loan dataset. Note that the learning of user-specific (personalized) sequence weights should decrease the time taken for pattern mining, because by eliminating all the user-irrelevant sequences (i.e. their weights are equal to zero) in SDB, we only need to handle a much smaller personalized hypothesis space consisting of sequences that are more relevant to the target users. On the contrary, without exploiting personalized sequence weights, sequential pattern mining algorithm will be performed inefficiently since it has to go through all the transactions in SDB.



(a) Shorter time taken to mine patterns (lower is better) due to the sequence weight learning.

(b) Total time taken (lower is better) for the sequence weight learning and the pattern mining.

Fig. 2. Improvement in time taken as a result of our proposed personalized learning framework

Figure 2(a) shows the time taken to mine sequential patterns over different minimum support values when there is no personalized sequence weight learning, versus with sequence weight learning using the different methods presented in Section 4.2. Indeed, our framework significantly reduced the time for the sequential pattern mining, especially with our Competence Score weighting method. Figure 2(b) compares the total time taken for both learning sequence weights and running the sequential pattern mining algorithm. Taking into account the sequence weight learning time, the user-independent learning method, which completely ignores the target users and computes sequence

weights based on their items' popularity, added significantly to the time taken, making it less attractive compared to sequential pattern mining without sequence weight learning. However, with reference to Figures 2(a) and 2(b), the extra time spent on the more user-specific weight learning methods (LCS, cosine similarity, Competence Score) is worthwhile since it greatly reduced the time used in the subsequent pattern mining step. More specifically, our proposed Competence Score method added only an average of 6.37 seconds for the weight learning but led to an average reduction of more than 20 seconds in pattern mining time. Considering that the learning of Competence Score and other methods to compute personalized sequence weights can be completed off-line, the time saving in sequential pattern mining due to our framework (Figure 2(a)) is even more significant.

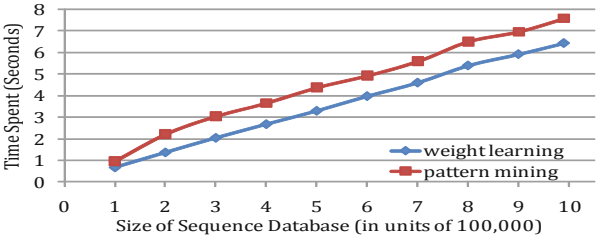


Fig. 3. The time spent for weight learning and pattern mining using the Competence Score method grows linearly with the size of sequence database (varying from 100,000 to 989,818 sequences).

Figure 3 shows the total time spent in learning the personalized sequence weights based on our proposed Competence Score, and in mining the frequent patterns from the personalized sequence database (with *minsup*=0.002). Both the weight learning and the pattern mining time increased in a linear manner as we increased the size of the database from 100,000 sequences to 989,818 sequences (the full size of the msnbc.com dataset), demonstrating the scalability of the proposed method for personalized recommendation. Further significant savings in weight learning time can be achieved by only considering those sequences in the database which have at least one common itemset with the target sequence (e.g., using the standard inverted index approach in search engines [35] so that sequences without any of the itemsets in the target sequence are pruned immediately).

The sequence weight learning methods in Section 4.2 use different relevance criteria (i.e., popularity, LCS, cosine similarity, and Competence Score) to compute personalized sequence weights. For each method, we evaluate the *quality* of the high-support sequential patterns in terms of how well they satisfy the corresponding criterion used to learn the sequence weights (similar results are observed for the top 5, 10, 15, 20 and 25 high-support patterns, and for *minsup* 0.002-0.010). We compare the results to the corresponding cases where personalized sequence weights are not used. The proposed Competence Score method improved pattern quality by five times, the LCS and cosine similarity methods improved pattern quality by three times, and the user-independent weight learning method based on item popularity also managed to improve the pattern

quality, albeit marginally. The results clearly show that, for each of the sequence weight learning methods, the relevance criterion used to compute the sequence weights has been effectively exploited to improve the quality of the resulting high-support patterns, such that the discovered patterns inherit the corresponding desirable relevance criterion.

The improvement in mining time and pattern quality from the item popularity-based weight learning method, which ignores the target user, is much less than our personalized methods (LCS, cosine similarity, and Competence Score) that learn user-specific sequence weights . This shows that even when using the same sequential pattern mining algorithm, the more user-specific sequence knowledge helps to generate high-support sequential patterns that are more relevant to the target users. This could explain why simply picking patterns to match the users *after* sequential pattern mining is inadequate [34,23]; since patterns that are meaningful for a user may not be frequent among the sequences for all users, directly using standard sequential pattern mining algorithm (without learning and exploiting sequence weights) would miss those *personalized* patterns which really match the users' preferences, indicating that it is crucial to integrate user relevant sequence knowledge into the pattern mining process.

5.3 Accuracy of the Next-Items Recommendation

We can now compare the different methods for next-items recommendation using both the msnbc.com and book-loan datasets. Using a standard *backlog* evaluation method, we partition each target user's test sequence into two portions: release only the earlier portion to the recommendation methods, and evaluate the accuracy by comparing the recommended items to the held-out portion. We need to experiment with sequences having sufficient items (at least ten items in our experiments). For each sequence, we release the first five items (known portion) and hold-out the remaining items as the ground-truth for evaluation. We allow each method to recommend up to ten items with the highest support values and evaluate results in terms of *recall*, *precision* (*precision@1*) and *F1-measure*, all of which are standard evaluation metrics for recommendation systems [12,23,34]. As there are test sequences (or cases) for which certain methods do not give any recommendation, we also measure the *applicability* of each method in terms of the percentage of test cases for which recommendations are given. We set the *minsup* for msnbc.com and book-loan as high as possible to 0.5 and 0.01, respectively, so that most of the less frequent patterns are effectively filtered off by the sequential pattern mining, while maintaining a minimum applicability of around 40% for the recommendation.

The results for the msnbc.com and book-loan datasets are presented in Table 2. The experimental results clearly demonstrate that the methods for learning of sequence weights which are more related to the target users can indeed yield significantly more accurate next-items recommendations. In particular, sequence knowledge learning using our proposed recommendation Competence Score in Section 4.2 produced the greatest improvement in performance among the competing methods; it provided recommendations for almost 100% of the test cases (a vast improvement in *applicability*), and significantly increased the recall, precision, F1-measure, and precison@1 (i.e., proportion of test cases where the top-1 recommended item is correct). Specifically, for the msnbc.com dataset, 70.6% of the test cases contained the first recommended item when Competence Score was used, compared to just around 40% for the competing

Table 2. Recommendation performance. Indicated in brackets are the % improvement over and above sequential pattern mining with no sequence weight learning (denoted as "no seq. weight").

(a) Performance on msnbc.com dataset.

	Applicability	**Recall**	**Precision**	**F1-measure**	**Precision@1**
no seq. weight	45.4%	0.178	0.313	0.227	0.375
popularity	45.4% (0%)	0.178 (0%)	0.313 (0%)	0.227 (0%)	0.375 (0%)
cosine sim.	49.1% (8%)	0.223 (25%)	0.342 (9%)	0.270 (19%)	0.404 (8%)
LCS similarity	52.3% (15%)	0.229 (29%)	0.366 (17%)	0.282 (24%)	0.435 (16%)
competence	100% (120%)	0.547 (207%)	0.502 (60%)	0.524 (131%)	0.706 (88%)

(b) Performance on book-loan dataset.

	Applicability	**Recall**	**Precision**	**F1-measure**	**Precision@1**
no seq. weight	39.5%	0.017	0.073	0.028	0.075
popularity	43% (9%)	0.024 (41%)	0.063 (-14%)	0.035 (25%)	0.07 (-7%)
LCS similarity	93% (135%)	0.069 (306%)	0.083 (14%)	0.075 (168%)	0.135 (80%)
cosine sim.	95% (141%)	0.074 (335%)	0.092 (26%)	0.082 (193%)	0.135 (80%)
competence	99% (151%)	0.077 (353%)	0.094 (29%)	0.085 (204%)	0.155 (107%)

methods. Likewise for the book-loan dataset, our proposed Competence Score method was able to outperform all the competing methods in terms of the various evaluation metrics. In particular, while the baseline method without any sequence weight learning managed a mere 4% accuracy in terms of predicting one or more of the next-books that were subsequently borrowed by the target users in the six-months period, our proposed personalized recommendation framework using the Competence Score method for sequence weight learning was able to achieve a significantly higher accuracy of 57%. The experimental results thus demonstrate that our personalized framework predicts future items reliably and can be used to automatically recommend next-items to target users.

6 Conclusions

Learning about the different importance of sequences can improve the sequential pattern mining-based next-items recommendation in real-world applications. We present a novel *personalized sequence weight learning and sequential pattern mining-based next-items recommendation framework* that learns every sequence's importance as weights, and then effectively exploits this additional sequence knowledge to improve the efficiency and the quality of learning in sequential pattern mining algorithms. Experimental results using real-world sequence datasets demonstrate that the proposed framework is highly effective in learning and exploiting the sequence knowledge for sequential pattern mining, and that this significantly improves the performance of the personalized next-items recommendation, especially with our novel Competence Score method.

The proposed framework can be readily applied for next-items recommendation in domains such as web mining, financial mining, and product/service consumption analysis, etc. An interesting future work is to explore sequence weighting methods which exploits user knowledge beyond what is present in the sequences, e.g., social influence

measures [10] and similarity in user vocabulary [28]. It is also interesting to investigate the combination of weight knowledge from multiple methods (i.e., early fusion) in a single recommender, and the efficacy of a recommender ensemble that inherits weight knowledge from many methods (i.e., late fusion). Other measures of recommendation performance like coverage, diversity and serendipity (e.g. [19]) can also be investigated. More generally, the benefits of personalized mining can be realized in data mining problems beyond sequential pattern mining (e.g. in classification, clustering); the question is how we can effectively personalize inputs to these algorithms for user-specific results.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of IEEE International Conference on Data Engineering (ICDE 1995), pp. 3–14 (1995)
2. Asuncion, A., Newman, D.: UCI ML Repository (2007), http://www.ics.uci.edu/~mlearn
3. Ayres, J., Gehrke, J., Yiu, T., Flannick, J.: Sequential pattern mining using a bitmap representation. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), pp. 429–435 (2002)
4. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12, 331–370 (2002)
5. Capelle, M., Masson, C., Boulicaut, J.-F.: Mining Frequent Sequential Patterns under a Similarity Constraint. In: Yin, H., Allinson, N.M., Freeman, R., Keane, J.A., Hubbard, S. (eds.) IDEAL 2002. LNCS, vol. 2412, pp. 1–6. Springer, Heidelberg (2002)
6. Chang, J.H.: Mining weighted sequential patterns in a sequence database with a time-interval weight. Knowledge-Based Systems (2010) Available Online
7. Chen, E., Cao, H., Li, Q., Qian, T.: Efficient strategies for tough aggregate constraint-based sequential pattern mining. Information Sciences 178(6), 1498–1518 (2008)
8. Cheng, H., Yan, X., Han, J.: IncSpan: Incremental mining of sequential patterns in large databases. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004), pp. 527–532 (2004)
9. Cheng, H., Yan, X., Han, J., Hsu, C.-W.: Discriminative Frequent Pattern Analysis for Effective Classification. In: Proceedings of IEEE International Conference on Data Engineering (ICDE 2007), pp. 716–725 (April 2007)
10. Goyal, A., Bonchi, F., Lakshmanan, L.V.S.: Learning influence probabilities in social networks. In: Proceedings of ACM International Conference on Web Search and Data Mining (WSDM 2010), pp. 241–250 (2010)
11. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., et al.: FreeSpan: Frequent pattern-projected sequential pattern mining. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2000), pp. 355–359 (2000)
12. Huang, C.-L., Huang, W.-L.: Handling sequential pattern decay: Developing a two-stage collaborative recommender system. ECRA 8(3), 117–129 (2009)
13. Huang, J.-W., Tseng, C.-Y., Ou, J.-C., Chen, M.-S.: A general model for sequential pattern mining with a progressive database. IEEE TKDE 20(9), 1153–1167 (2008)
14. Kum, H.-C., Pei, J., Wang, W., Duncan, D.: ApproxMAP: Approximate mining of consensus sequential patterns. In: Proceedings of SIAM Intl. Conf. on Data Mining, pp. 311–315 (2003)
15. Li, C., Lu, Y.: Similarity measurement of web sessions by sequence alignment. In: Procs. of IFIP Intl. Conf. on Network and Parallel Comp. Workshops (NPC 2007), pp. 716–720 (2007)
16. Lin, M.-Y., Hsueh, S.-C., Chang, C.-W.: Fast discovery of sequential patterns in large databases using effective time-indexing. Information Sciences 178(22), 4228–4245 (2008)

17. Lo, S.: Binary prediction based on weighted sequential mining method. In: Proceedings of International Conference on Web Intelligence (WI 2005), pp. 755–761 (2005)
18. Mitchell, T.: Machine Learning. McGraw Hill (1997)
19. Onuma, K., Tong, H., Faloutsos, C.: TANGENT: A novel, 'surprise me', recommendation algorithm. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009), pp. 657–665 (2009)
20. Pei, J., Han, J., Mortazavi-Asl, B., et al.: Mining sequential patterns by pattern-growth: The PrefixSpan approach. IEEE TKDE 16(11), 1424–1440 (2004)
21. Pei, J., Han, J., Wang, W.: Mining sequential patterns with constraints in large databases. In: Procs. of ACM Intl. Conf. on Info. and Knowl. Management (CIKM 2002), pp. 18–25 (2002)
22. Pei, J., Fu, A.W.-C., Lin, X., Wang, H.: Computing compressed multidimensional skyline cubes efficiently. In: Procs. of IEEE Intl. Conf. on Data Eng. (ICDE 2007), pp. 96–105 (2007)
23. Pyo, S., Kim, E., Kim, M.: Automatic recommendation of (IP)TV program schedules using sequential pattern mining. In: Adjunct Proceedings of EuroITV 2009, pp. 50–53 (2009)
24. Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: Factorizing personalized Markov chains for next-basket recommendation. In: Proceedings of International Conference on World Wide Web (WWW 2010), pp. 811–820 (2010)
25. Resnick, P., Varian, H.R.: Recommender Systems. Comms. of the ACM 40(3), 56–58 (1997)
26. Salton, G.: Automatic text processing: The transformation, analysis, and retrieval of information by computer. Addison-Wesley Longman Publishing (1989)
27. Saneifar, H., Bringay, S., Laurent, A., Teisseire, M.: S^2MP: Similarity measure for sequential patterns. In: Procs. of Australasian Data Mining Conf. (AusDM 2008), pp. 95–104 (2008)
28. Schifanella, R., Barrat, A., Cattuto, C., et al.: Folks in folksonomies: Social link prediction from shared metadata. In: Proceedings of ACM International Conference on Web Search and Data Mining (WSDM 2010), pp. 271–280 (2010)
29. Sequeira, K., Zaki, M.: Admit: Anomaly-based data mining for intrusions. In: Proceedings of ACM Intl. Conf. on Knowledge Discovery and Data Mining (KDD 2002), pp. 386–395 (2002)
30. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 3–17. Springer, Heidelberg (1996)
31. Torrey, L., Shavlik, J.: Transfer Learning. In: Handbook of Research on Machine Learning Applications. IGI Global (2009)
32. Yun, U.: A new framework for detecting weighted sequential patterns in large seq. databases. Knowledge-Based Systems 21, 110–122 (2008)
33. Zaki, M.J.: SPADE: An efficient algorithm for mining frequent sequences. Machine Learning 42(1/2), 31–60 (2001)
34. Zhou, B., Hui, S.C., Chang, K.: An intelligent recommender system using sequential web access patterns. In: Procs. of IEEE Conf. on Cybernetics and Intell. Sys., pp. 393–398 (2004)
35. Zobel, J., Moffat, A.: Inverted files for text search engine. ACM Comp. Surveys 38(2) (2006)