An Empirical Study of Extracting Multidimensional Sequential Rules for Personalization and Recommendation in Online Commerce

Arthur Pitman
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria
arthur.pitman@uni-klu.ac.at

Markus Zanker
Alpen-Adria-Universität Klagenfurt
Klagenfurt, Austria
markus.zanker@uni-klu.ac.at

ABSTRACT

The application of web mining to personalization has a long tradition in electronic commerce research. In this empirical study we focus specifically on mining sequential navigation patterns from weblogs and thoroughly compare different design variants for making personalized suggestions to users. In particular we concentrate on the impact of additional product knowledge like item characteristics, different properties of the sequential pattern mining process such as *closure* as well as rule quality metrics such as support, confidence and lift, and evaluate the recommender's accuracy by experimenting on historical web sessions.

This paper therefore firstly demonstrates how state of the art sequence mining algorithms such as PrefixSpan and BIDE may be adapted to the specific problem of extracting sequential rules from e-commerce weblogs. Furthermore, in order to compact the resulting rule set, the Δ-closed criteria is proposed as a logical extention to closed and maximal frequent patterns to eliminate spurious sequences. Finally, our experimental findings show that using multidimensional sequential patterns and the lift metric for weighting personalization rules can boost recall to 28% of all actual purchase transactions when using only short navigational sequences.

Keywords

business informatics in commerce, recommender systems, sequential pattern mining

1. INTRODUCTION

Understanding the navigational patterns of prospective customers in online commerce and exploiting them in order to provision personalized services was first considered over a decade ago [20]. Several empirical studies have shown that value-added search mechanisms like recommendation systems can for instance lead to increased shopping enjoyment and intention to return [11, 12]. Therefore, we evaluate the effectiveness of mining navigational user patterns in the sense of Kohavi [10] to develop workable recommendation mechanisms with the aim of increasing customer satisfaction and conversion rates.

Recommender systems are usually described as software agents that predict which items a user will probably like based on elicited preferences and interests [25]. Typically, these preferences are represented by ratings where users either explicitly assign a value expressing their opinion (e.g. the degree how much they like a movie) or implicit actions such as purchases are interpreted as positive unary ratings. However, this research focuses only on very shallow user profiles obtained from the navigational actions of anonymous web sessions, typically consisting of a series of menu navigation and item view events. The system extracted web sessions that led to add-to-basket actions and tried to predict the item selected based on the navigational history since the start of the session.

Our evaluation proceeded as follows: we first extracted sequential patterns using a customized version of a PrefixSpan [17] based algorithm and then exploited these sequences to build rule sets for personalization and recommendation. We then explored different recommender design variants by experimentally comparing how an additional product information, the amount of rules as well as different rule weights such as support, confidence and lift impacted the accuracy of the recommendation lists produced. Thus, in addition to the discussion, the paper also contributes empirical findings from an offline evaluation comparing these different variants on a dataset extracted from the weblogs of a leading online nutritional supplements retailer.

In practice, the value of rule-based recommender systems is often limited by the level of detail modeled in the source data. One approach is to extend typical item-based data with other multi-dimensional attributes [18]. Indeed, such information is often readily available as products are typically organized into a hierarchy or categorized using tags. For example a simple sequence of view and purchase actions, such as $\langle$*views product A, buys product B*$\rangle$ could be extended to $\langle$*(views milk, views skim milk, views product A), (buys bread, buys brown bread, buys product B)*$\rangle$, allowing more subtle relationships to be exposed involving the higher level product categories of *bread* and *milk*.

10^{th} International Conference on Wirtschaftsinformatik,
16^{th} – 18^{th} February 2011, Zurich, Switzerland

In our evaluation we demonstrate how standard sequential data can be augmented to create multidimensional sequences and explore the associated positive impact on performance. On the downside, multidimensional sequences add enormously complexity due to an exploding amount of additional patterns due to combinatory. We address this issue by examining specific subsets of frequent patterns notably closed, maximal and Δ-closed, where the latter is an intermediate subset of the former two. We show that we are able to identify smaller subsets of frequent patterns that perform comparably to the larger superset. Third, we also vary the weights of extracted rules among support, confidence, support × confidence and lift measures and our results indicate that the lift metric has very favorable properties when applied to multidimensional data.

Concluding, the worth of this paper lies in its particular focus on evaluating how multidimensional sequential patterns can be exploited for identifying items to recommend and, to the best of authors' knowledge, no comparable work has been published so far.

Following this section, in Section 1 we present a brief survey of related work. Section 2 defines the problem of sequence mining and examines the basis of our recommender, sequential pattern mining. In Section 3, we describe the specifics of our recommender. Following the introduction of our experimental methodology in Section 4, the recommender is evaluated in Section 5. Finally, our conclusions are given in Section 6.

sectionOverview of Related Work Since Agrawal and Srikant first investigated sequential pattern mining in [1], a significant body of research has focused on improvements in terms of time and space complexity. For example, algorithms such as GSP [21] and PrefixSpan [17] both attempt to reduce the search space for finding frequent sequences. Other algorithms, such as PRISM [6], attempt to reduce the effort required to determine the support of a sequence. The next wave of innovation (e.g. [26] and [23]) dealt with reducing the result set to closed sequences, thus overcoming the potentially exponential number of frequent sequences mined under certain conditions. The specifics of the PrefixSpan [17] and BIDE [23] algorithms are examined in Section 2. Further works have also focused on domain specific challenges, such as mining the top-k closed sequential patterns [22] and temporal patterns [16]. The area of multi-dimensional sequence mining, of particular interest to this work, has also seen significant research. Such datasets may either be handled using a domain-specific or general approach, as was examined in [18]. These systems may also be optimized for hierarchical data (e.g. taxonomies) [19].

While the aforementioned works focus mainly on performance issues, we are more interested in the application of sequential pattern mining for web personalization comparable to [13] or [14, 15] who explored association rule mining from weblogs for identifying items to recommend. However, as pure association rule mining disregards temporal relationships in the data, efficient sequential pattern mining is more suitable. Therefore, Büchner et al. [4] did first approaches towards identifying sequential patterns from weblogs. Berendt and Spiliopoulou's approach [2] is more similar to our work as they also used templates to constrain the search to patterns with desirable characteristics like we did here. Eirinaki and Michalis [5] elaborated on the various efforts taken to exploit web usage mining for web personalization, however our approach particularly focuses on multidimensional sequential patterns and comparing design variants based on their predictive accuracy, which has not yet been done.

2. SEQUENTIAL PATTERN MINING

2.1 Problem Definition

Let $s = \langle e_1 e_2 \dots e_l \rangle$ be a sequence of length l, where each element $e_j = (x_1 x_2 \dots x_m)$ is a subset of the set of all items $I = \{i_1, i_2, \dots, i_n\}$. Building upon the work of [23], this paper uses the following terminology:

- A sequence database SDB is a set of tuples of the form $\langle id, s \rangle$, where s is a sequence identified by id.
- A sequence $a = \langle a_1 a_2 \dots a_n \rangle$ is a subsequence of a sequence $b = \langle b_1 b_2 \dots b_m \rangle$ (i.e. b contains a or is a supersequence of a) if and only if integers $1 \le j_1 < j_2 < \dots < j_n \le m$ exist such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \dots, a_n \subseteq b_{j_n}$. Furthermore, a sequence b supports a sequence a if b contains a (or a is a subsequence of b).
- The support of a sequence a with respect to a sequence database SDB is defined as the number of sequences in SDB that contain a. We write this using the notation a:*support*(a). Furthermore, a is labeled as frequent if *support* $(a) \ge min\_support$. We also refer to frequent sequences as sequential patterns.

In general, the task of a sequential pattern miner is to find the set of all frequent sequential patterns, S_f, for a given sequence database.

EXAMPLE 1 (Sequential Pattern Mining). Consider the sample sequence database shown in Table 1. For simplicity it is a simple sequence database (SSD), i.e. each element only contains a single item, however the same arguments apply to general sequence databases (GSDs). The sequence database SDB contains four sequences constructed from a total of four items. If, for example, sequential patterns are mined with minimum support $min\_support = 2$ then a total 24 sequences are generated: A:4, AC:3, ACD:2, AD:4, ADC:2, B:3, BA:3, BAD:3, BC:2, BCD:2, BD:3, C:3, CA:2, CAD:2, CD:3, D:4, DA:2, DAD:2, DB:2, DBA:2, $DBAD$:2, DBD:2, DC:2 and DD:2.

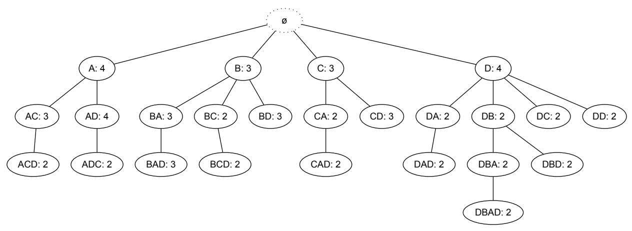
2.2 Generalized Sequence Mining

The Generalized Sequential Pattern (GSP) algorithm is probably the most straightforward approach to sequence mining, which iteratively extends existing sequences to generate new sequences [21]. Despite its simplicity, it is inefficient as it requires the database to be scanned multiple times. Furthermore, many candidate sequences will not meet minimum support requirements and are subsequently discarded.

Table 1: Example sequence database SDB

ID	Sequence
1	C A D C
2	A D B C A D
3	D B A D
4	B A C D

Figure 1: The tree produced by applying PrefixSpan to SDB



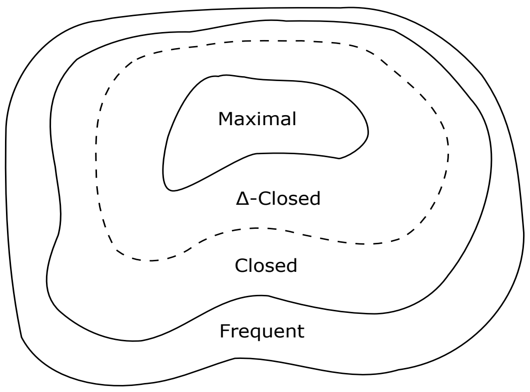
In [17], Pei et al. develop a method for traversing the tree of possible frequent sequential patterns in a more efficient manner. In contrast to other a-priori sequential pattern mining algorithms, such as GSP, which follow a generate-and-test approach, PrefixSpan grows patterns directly by recursively dividing the original sequence database and only considering sequential patterns that satisfy the minimum support criterion.

With every recursive call to the algorithm, the current pattern is extended by one item, either by appending a new element (sequential extension or $S-extension$) or by adding to the last element (itemset extension or $I-extension$). A new (smaller) sequence database is then created for each new pattern that satisfies the minimum support criterion.

As PrefixSpan grows patterns from left to right, the new sequence database only needs to contain the elements of each sequence including and appearing after the last element of the first occurrence of the new pattern. To remove the overhead of actually copying these elements with each recursion step, Pei et al. suggest using pseudo-projected databases. Each pseudo-projected database is a collection of projected sequences, tuples in the form (id, offset) consisting of a sequence id and a projection point offset pointing to the last element of the first occurrence of the new pattern in the original sequence. In this work, we group both the pseudo-projected database and the current sequential pattern into a *projection*, an object which effectively encapsulates the recursive state of PrefixSpan.

Example 2 (PrefixSpan). As illustrated in Figure 1, PrefixSpan generates a tree with each node corresponding to a projection. Due to the recursive nature of the algorithm, nodes are discovered in a depth-first fashion, while children are typically visited in lexical order.

Figure 2: Relationship between sequence subsets



2.3 Mining Subsets

Despite being an exhaustive set, S_f can hardly be considered compact. In general, if sup_{min} is reduced or the database size or sequence length is increased, the number of frequent sequential patterns tends to grow exponentially. For example, if a single sequence containing 100 single item elements is mined with $sup_{min} = 1$, $2^{100} - 1$ sequences and thus projection nodes (in the case of PrefixSpan) will be produced. One possibility is to only consider a subset of all frequent sequences. Previous research, e.g. [27] and [7], has focused in particular on the *closed* and *maximal* subsets of frequent sequences as defined in the following subsections. As outlined in Figure 2, we will also introduce an intermediate subset, namely the subset of Δ-*closed* sequences that combines the favorable properties of both the closed and maximal frequent sequences by reducing the quantity of sequential patterns with minimal impact on their information content.

2.4 Closed Sequence Mining

The idea of reducing the entire set of sequences to closed sequences offers a possibility to improve compactness and thus improve efficiency while maintaining completeness [27] [26]. Closed sequences are defined analogously to the idea of closed itemsets, i.e. a sequence s_a is closed if and only if no supersequence s_b of s_a exists with the same support ($\nexists s_b$ such that $s_a \subset s_b$ where $support(s_a) = support(s_b)$). Formally, the set of closed frequent sequences S_{f_c} is the most minimal subset of S_f that maintains support information about all possible subseqences. Thus, if desired, all frequent sequences can be enumerated from the set of closed frequent sequences.

Example 3 (Closed Sequence Mining). Figure 3 extends Figure 1 by highlighting the closed nodes from our running example, AC:3, ACD:2, AD:4, ADC:2, BAD:3, BCD:2, CAD:2, CD:3 and $DBAD$:2. Note that only 9 of the 24 frequent sequences are closed.

Unfortunately, determining if a sequence is closed is significantly more complex than for an itemset in association rule

Figure 3: Closed sequences of SDB

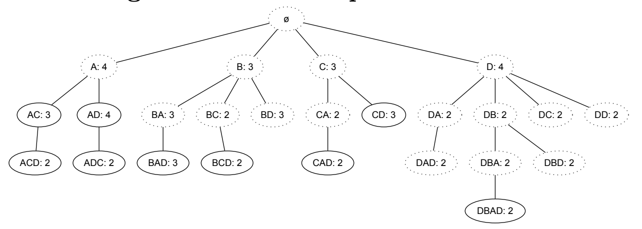
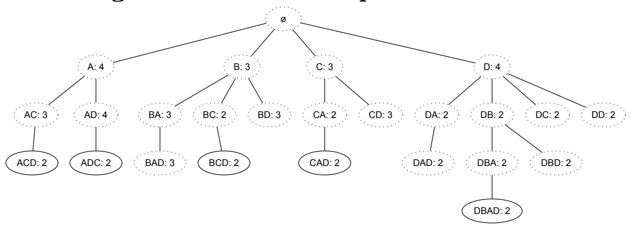


Figure 4: Maximal sequences of SDB



mining due to the constraints imposed by sequential ordering. Popular approaches include maintaining an optimistic list of closed sequences which is periodically pruned (similar to [26] and [27]) and exhaustively checking sequences during enumeration to produce a final set of closed sequences (e.g. [23]).

Bi-Directional Extension (BIDE) closure checking determines closure by re-examining the source sequences that give support to a pattern [23]. It carries the distinct advantage that a list of potentially closed (or actually closed) sequences must not be maintained throughout the sequence mining process. Specifically, if no forward or backward extension with the same support exists for the current pattern, then the current pattern is a closed sequence.

Forward extension checking is straightforward and corresponds to examining the projections generated during PrefixSpan pattern growth (i.e. child projections) and determining if any exhibit the same support as the current projection.

Backwards extension checking is less trivial and requires determining if the current sequence may be extended by inserting an item into an existing element or as a new element prior to end of the sequence to produce a sequence with the same support.

Although confirming the presence of a backwards extension requires rescanning the entire sequence database of the current projection, in cases where a sequence does not permit such an extension, the scan may be aborted early (referred to as ScanSkip in [23]). In general, however, the cost of using BIDE closure is proportional to the number of sequences in the projection and their average length.

2.5 Closed Pruning Techniques

Although closed sequences may be employed to reduce the size of the result set, they do not improve mining efficiency: all nodes of the projection tree must still be traversed. As a result, pruning methods have been developed to complement PrefixSpan to determine in advance if a pattern or any of its supersequences will be found to be closed. Obviously, if this is not the case, the corresponding node must not be visited.

The requirements of pruning algorithms differ from closure checking algorithms in that while the latter must correctly classify sequences as open or closed, pruning algorithms may conservatively decide to visit a node with no impact on the correctness of the output of the algorithm. Furthermore, pruning should only be attempted if the cost of pruning is less than that of actually visiting the node.

The BackScan pruning method is described in [23]. Similar to BIDE closure checking, it considers pruning a node by examining if it and its descendants will be found to be subsequences of sequential patterns with the same support generated elsewhere in the projection tree.

Importantly, BackScan pruning differs from the backward extension detection of BIDE closure checking in that while the latter only requires that an insertion is supported by one of the occurrences of a projection's pattern within each supporting sequence, BackScan requires that such an insertion is supported by each occurrence.

Example 4 (Closed Pruning). Reexamining Figure 3, it is clear that some paths do not lead to closed sequences. For example, neither DA:2 nor its descendent DAD:2 is closed due to $DBAD$:2 and thus must not be visited. In other cases, pruning may prevent nodes from being expanded. For example, BD:3 although visited must not be expanded due to the presence of BAD:3. Similar arguments apply for DBD:2, DC:2 and DD:2.

2.6 Maximal Sequence Mining

For some applications the set of closed sequences S_{f_c} may still not be compact enough. For example, it may be too large for domain experts to comprehend or for use in a recommender system. One possibility is to use the set of maximal sequences which provides an even more compact summary of the frequent sequences. Formally, a sequences is maximal if and only if all supersequences are infrequent. In comparison to the set of closed sequences, maximal sequences provide no information about the support of subsequences and thus cannot be used to reconstruct the entire set of frequent sequences.

Example 5 (Maximal Sequence Mining). As depicted in Figure 4, after filtering S_{f_c} only 5 sequences remain as maximal sequences: ACD:2, ADC:2, BCD:2, CAD:2 and $DBAD$:2. This information is however insufficient to determine the support of for example the sequence B. Instead we may only estimate its support as lying between the nearest maximal supersequence and size of the database, i.e. $support(BCD) \leq support(B) \leq |SDB|$.

Despite the fact that efficient algorithms are available for mining maximal itemsets [7] [8], to the best of our knowl-

Figure 5: Δ-closed sequence algorithm

```
1:  procedure Δ-Close($S_{f_c}$, Δ)
2:      $S \leftarrow OrderBySupportAscending(S_{f_c})$
3:      $i \leftarrow 1$
4:      while $i \le Count(S)$ do
5:          $b \leftarrow Support(S_i) + \Delta$
6:          $j \leftarrow i + 1$
7:          while $j \le Count(S)$ do
8:              if $Contains(S_i, S_j)$ and
9:               $Support(S_j) \le b$ then
10:                 $S \leftarrow S - \{S_j\}$
11:             else
12:                 $j \leftarrow j + 1$
13:             end if
14:         end while
15:         $i \leftarrow i + 1$
16:     end while
17:     return $S$
18: end procedure
```

edge no such algorithms are available for maximal sequences. The set of maximal sequences may however be obtained by post processing S_{f_c} by removing any sequence that is a subsequence on another closed sequence.

2.7 Δ-Closed Sequence Mining

While the compactness of maximal sequences is appealing the information loss is often too extreme. As a result we introduce the concept of Δ-*closed* sequences that provides a compromise between the two. Formally, a sequence s_j is Δ-closed if and only if no supersequence s_i exists such that $support(s_j) \le support(s_i) + \Delta$. This definition has two important consequences:

1. The set of 0-closed sequences for a sequence database is equivalent to its set of closed sequences as defined in Subsection 2.4.

2. Test set of d-closed sequences for a sequence database SDB, where $d = |SDB|$, is equivalent to its set of maximal sequences as defined in Subsection 2.6.

Figure 5 outlines a simple method for deducing Δ-closed sequences from closed sequences. The algorithm examines sequences ordered by increasing support, and for each sequence s_i removes any other subsequence s_j which offers an increase in support less than or equal to Δ. As part of our future work, we hope to integrate this concept directly into the sequence mining algorithm to further increase efficiency.

Example 6 (Δ-Closed Sequence Mining). Figure 6 illustrates the function of the Δ-closed sequence algorithm for $\Delta = 1$. Starting from the bottom of the PrefixSpan tree (i.e. those sequences with the lowest support), the algorithm removes other subsequences which offer a support increase less than or equal to Δ. Considering the lexical order of the sequences, the sequence $ACD:2$ causes AC:3 to be removed, BCD:2 causes CD:3 to be removed and $DBAD$:2 causes BAD:3 to be removed. The analysis continues for sequences with a support of 3. Notice that as BAD:3 has already been removed, AD:4 remains in the result set.

Figure 6: 1-closed sequences of SDB

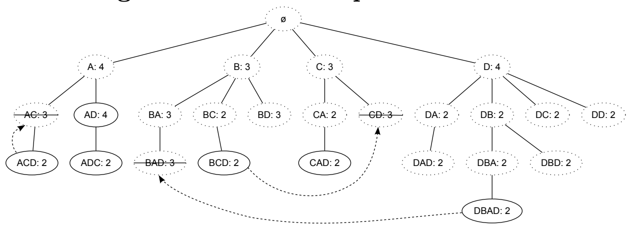
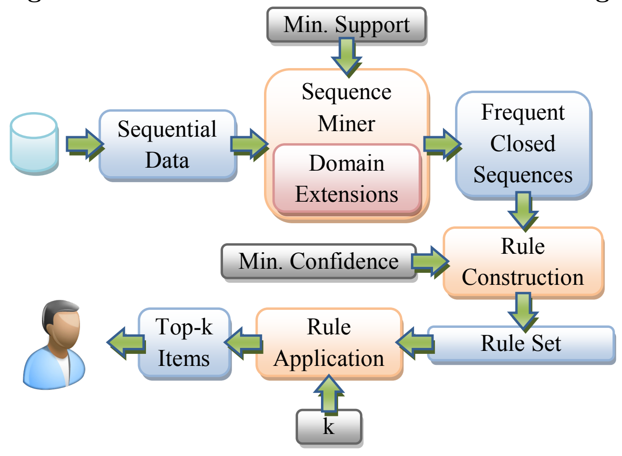


Figure 7: An overview of the recommender design



3. RECOMMENDER DESIGN

Our recommender was designed specifically with the intention of predicting a user's first add-to-basket action following a series of navigational or view actions, collectively referred to as *leading events*. The recommendation process can be decomposed into three stages, mirroring the design proposed in [13]: the sequential pattern mining stage, the rule construction stage and the rule application stage. A overview of the recommender design is illustrated in Figure 7.

3.1 Sequential Pattern Mining Stage

Rather than naïvely mining all relevant sequences from the learning set (i.e. the frequent, closed, Δ-closed or maximal sequences) and then filtering the results, the sequence miner was modified to only mine sequences ending with an add-to-basket event. This was immensely important as an initial analysis indicated that mining the entire set of sequences would be extremely expensive and thus render sequence mining at sufficiently low minimum supports infeasible.

As PrefixSpan builds sequences from left to right in a tree-like fashion mine, search space pruning is most effective near the root. As a result, the input sequences were reversed and a restriction that all sequences began with an add-to-basket event was enforced, drastically reducing the search space. Modifications also had to be made to both the BIDE closure and BackScan pruning algorithms to prevent any backwards extensions prior to the first item preventing such a sequence from being considered closed or inadvertently pruned. Following the mining process, the sequences were once again reversed.

In addition, limiting the maximum number of leading events improved the performance of the sequence miner, effectively limiting its maximum search depth.

3.2 Rule Construction Stage

Following the sequential pattern mining stage, the rule construction stage converted each sequence s in S_{f_c} into a sequential rule r in the form $r_l \rightarrow r_r$. As each sequence found in the previous stage consisted of a number leading events followed by an add-to-basket event this was quite straight forward: the left hand side of the rule r_l is simply the subset of leading events from s, while the right hand side r_r is the add-to-basket event. Optionally, in particular in situations where the size of the rule set must be minimized, a candidate rule r may be excluded from a rule set R when $confidence(r) < confidence_{min}$. Formally a rule's confidence is defined as (Equation 1):

$$Confidence(r_l \rightarrow r_r) = \frac{support(r_l \cup r_r)}{support(r_l)} \qquad (1)$$

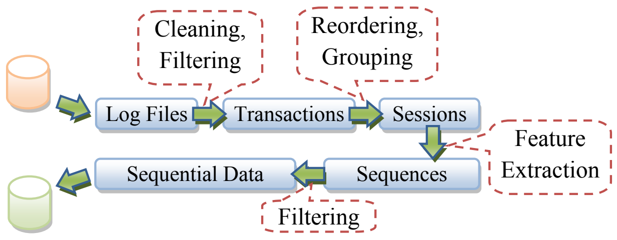
3.3 Rule Application Stage

The rule application stage utilizes the rule set R to produce a product ranking for a leading event sequence, s_l, comparable to the procedure outlined in [13]. For each rule $r \in R$, the left hand side r_l is compared against s_l and in the case that s_l contains r_l, the rule is activated and increases the ranking of the product or product categories specified by the right hand side r_r. The magnitude of the increase is typically given by a weight derived from the rule itself. We tested a number of weights, namely the rule's $relativesupport$ (Equation 2), $confidence$ (Equation 1), $relativesupport \cdot confidence$ [13] and *lift* (or *interest*)[3] for our recommender. Lift, whose equation is given in Equation 3, addresses the fact that high confidence rules may be misleading as confidence ignores the support of rule's consequent. Higher lift values indicate that the antecedent and consequent are not statistically independent and hence some relationship exists between them. Importantly, lift does not indicate a causal relationship but rather measures co-occurrence. Finally, the top-k products from the resulting ranking are selected as the output of the recommender as described by [13].

$$RelativeSupport(r_l \rightarrow r_r) = \frac{support(r_l \cup r_r)}{|SDB|} \qquad (2)$$

$$Lift(r_l \rightarrow r_r) = \frac{support(r_l \cup r_r)}{support(r_l) \cdot support(r_r)} \qquad (3)$$

Figure 8: Outline of data processing



4. METHODOLOGY

The datasets used for this evaluation were derived from the *OnlineShop*[1] click stream. This click stream, collected from a leading online supplier of nutritional supplements over two six month periods, between October 2008 and March 2009, and December 2009 and May 2010, contains HTTP requests from over 550,000 sessions. This was processed to create two datasets, namely *OnlineShop* and *OnlineShopMD* (a multidimensional version). Both datasets are currently publicly available online in an anonymized form at `http://sequin.codeplex.com/releases`.

4.1 Data processing

Prior to conducting the evaluation, the source data was processed as outlined in Figure 8, comparable to [15]. Working with the original Apache web server log files, the HTTP request records were cleaned and transformed into transactions. Records originating from search engine bots as well as image requests were removed. In a second step, the transactions were grouped into sessions based on the request's IP address and user agent string, thus identifying unique users. Furthermore the time between requests was considered: a new session was created in situations where more than 30 minutes elapsed between requests. Following this, the sessions were subjected to graph-based feature extraction to create sequences of events. In this case, we were interested in the browsing and buying actions of the customers and extracted events corresponding to product category view, product view and add-to-basket actions.

Finally, some additional filtering was applied to the sequences as, for purposes of our evaluation, only the subsequences leading up to and including the first add-to-basket event were relevant. Consequently, each sequence contained a series of view events and exactly one add-to-basket event. Moreover, product views directly before an associated add-to-basket event were removed to prevent erroneous rule generation as the site only permits products to be added to the shopping basket from a product view page. Sequences without at least two events were excluded as they were unsuitable for recommendation purposes.

4.2 Dataset Properties

The processed *OnlineShop* dataset contained 3,506 sequences with an average of 5.8 events per sequence (4.8 view events followed by an add-to-basket). As this dataset was single dimensional, each event contained one item. The multidimen-

[1]The identity of the online shop has been anonymized

sional *OnlineShopMD* dataset was created by reprocessing the *OnlineShop* dataset and inserting additional items into the events to represent their attributes. Each category view, product view or add-to-basket was expanded with items representing an item's position in the site's product hierarchy as obtained by crawling the website. The crawler recorded attributes such as product category, brand and price class, as well as allergy information and ingredients by traversing the site's menus and extracting key fields. In total, 175 additional unique items were added to the dataset. Although the expansion increased the average number of items per event to 4.7 items, the properties of this dataset otherwise remained the same.

4.3 Experimental Technique

Throughout the evaluation, the recommender's performance was measured using ten-fold cross validation [24] to ensure the general validity of results. The learning and testing phases were repeated 10 times using a different 10% of the source sequences for testing. In each instance the remaining 90% of the sequences were used for sequence mining to derive the rule set used for testing. During learning the recommender had access to both the lead events and the add-to-basket events, while during the testing phase the add-to-basket events were withheld.

Rules were matched with against a fixed number leading events (events prior to the add-to-basket event) and used to generate a recommendation set with the aim of predicting the sole withheld add-to-basket event. A *hit* signifies a successful recommendation, i.e. the add-to-basket event is contained within the recommendation set. Formally, the accuracy of a recommender can be defined in terms of recall and precision, as well as their harmonic mean, the $F1$ metric [9] (Equations 4, 5 and 6). In this case, as a maximum of one hit can be produced by each sequence, recall is defined as the proportion of sequences in which the recommendation set contained the hidden add-to-basket event. Similarly, as at most one hit is possible for each recommendation due to the experimental design, precision is proportional to the size of the recommendation set, k. Hence, in our evaluation we report only recall figures.

$$Recall = \frac{\sum hits}{\#sequences} \tag{4}$$

$$Precision = \frac{\sum hits}{\#sequences \cdot k} \tag{5}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6}$$

5. EVALUATION

The objective of our evaluation is to comparatively analyze the predictive accuracy of different sequential pattern mining variants and design variants of recommendation approaches and test the following hypotheses:

1. **(H1):** The impact of multidimensionality introduced by product knowledge: We hypothesize that additional semantic information about the recommendable items should increase the predictive accuracy of the recommendation approach.
2. **(H2):** Lift metric: Using the lift metric for weighting sequential rules should lead to improved accuracy results on multidimensional data.
3. **(H3):** The impact of Δ-closed frequent patterns: We assume that the Δ-closed concept can be utilized to filter sequential patterns without compromising on the accuracy of the recommendations generated.

5.1 Experimental Setup

To limit the complexity of the rule mining, environmental parameters of the sequence miner and rule generation component were held constant throughout the evaluation. Specially, we applied a minimum support of 10 sequences (i.e. a relative minimum support of 0.3%) and did not restrict the minimum confidence of the rules generated. Furthermore, we limited the maximum number of leading events to five view events prior to the add-to-basket in order to ensure that the sequences represented shallow user models. In addition, the system produced recommendation for five items and hence precision was exactly one-fifth of recall.

5.2 Impact of Multidimensionality

In a first step, we mined only single dimensional sequential rules and examine the resulting recommender recall. As outlined in Table 2, the recommender achieved a recall of approximately 16% which we subsequently use as a baseline. Rel. support $\times$ confidence performed the best out of the metrics, supporting the findings of [13].

We subsequently turned our attention to the effect of introducing multidimensionality into rule antecedents. Table 3 lists the performance of different recommender variants for a variety of sequence subsets. Clearly, the additional multidimensional information improved accuracy, increasing recall values to around 25%. Interestingly, despite the explosion of rules (approximately 2000$\times$) produced by using the complete set of frequent sequences, none of these recommender variants performed significantly better than those based on either the closed or maximal frequent sequence subsets. In addition, it can be observed that relative support lags behind the three other weighting metrics.

Following this we introduced further multidimensionality into the rule consequents, producing rules that affect the rankings of not only items but also item categories. The results are presented in Table 4. For complexity reasons, we were unable to mine and utilize the entire set of frequent sequences and were instead limited to using the closed and maximal subsets. Examining the increase in rule counts, it is clear that the increase resulted from the large number of categories, effectively reducing rules with product consequents to less than one percent of the rule base. Not surprisingly, recall declined sharply for most weighting metrics despite the extra information. As hypothesized in H2, the lift metric was more resilient, maintaining recalls of between 21 and 22%, being better able to identify statistically independent relationships.

Table 2: Recommender recall for single dimensional sequential rules

Sequence Subset	Rule Count	Rel. Support*	Confidence*	Rel. Support × Confidence*	Lift*
Frequent	76.3	16.3% ± 1.4%	16.1% ± 1.3%	16.3% ± 1.1%	14.8% ± 1.0%
Closed	76.3	16.1% ± 1.4%	16.0% ± 1.2%	16.3% ± 1.2%	14.7% ± 1.0%
Maximal	64.7	13.4% ± 1.0%	14.5% ± 0.8%	14.5% ± 0.9%	13.3% ± 0.9%

* 95% confidence intervals given

Table 3: Recommender recall for sequential rules with multidimensional antecedents

Sequence Subset	Rule Count	Rel. Support*	Confidence*	Rel. Support × Confidence*	Lift*
Frequent	1901987.3	22.8% ± 1.5%	24.6% ± 1.5%	23.7% ± 1.5%	24.0% ± 1.3%
Closed	891.1	21.4% ± 1.6%	25.3% ± 1.6%	23.8% ± 1.6%	24.9% ± 1.7%
Maximal	325.7	17.8% ± 1.4%	22.0% ± 1.7%	22.0% ± 1.6%	21.1% ± 1.3%

* 95% confidence intervals given

Table 4: Recommender recall for sequential rules with multidimensional antecedents and consequents

Sequence Subset	Rule Count	Rel. Support*	Confidence*	Rel. Support × Confidence*	Lift*
Closed	158210.9	11.8% ± 1.5%	19.0% ± 1.6%	17.4% ± 1.5%	22.0% ± 1.4%
Maximal	37351.6	6.4% ± 1.5%	14.7% ± 1.3%	14.6% ± 1.4%	21.1% ± 1.5%

* 95% confidence intervals given

Based on these findings we then considered the possibility of treating rules with category consequents as weaker rules and explored the effect of applying an additional weighting factor. As depicted in Figure 9, recall increased for smaller weights. A peak recall value of 28% was obtained for a weight of $\frac{1}{30}$.

Thus the findings supported both hypotheses H1 and H2.

5.3 Performance of Δ-Closed Frequent Sequences

Having observed that introducing multidimensional information into rule antecedents and consequents greatly increases the overall rule count, we considered the possibility of utilizing a reduced subset of frequent sequences. Figure 10 outlines the continuum of different sets of Δ-closed frequent sequences between the two extremes of closed and maximal frequent sequence subsets. Clearly it is possible to reduce the rule set by a factor of two while still preserving reasonable recommendation accuracy of around 28%, thus supporting Hypothesis H3.

6. CONCLUSIONS

In this paper we have presented an empirical study of different design variants of sequential pattern mining for recommending items of interest to shallow user profiles. Using a historical dataset obtained from a leading e-tailer of nutritional supplements, we demonstrated how integrating multidimensional product knowledge can be used to increase recommender recall. Furthermore, we introduced the concept of Δ-closed frequent sequences and showed how this can be applied to reduce the volume of patterns generated without compromising on accuracy of recommender results. The best recommender configuration, utilizing lift, achieved a remarkable recall of 28% for short user navigation sequences of less than five view events.

7. SOFTWARE

The software for the evaluation was implemented using the *Sequin* library, an open-source sequence mining library written in C#. Created specifically to support the needs of web mining for personalization, *Sequin* provides classes for web server log processing, session reconstruction, a customizable implementation of the BIDE algorithm and a platform for performing evaluations. The library, together with source code and documentation, is available from `http://sequin.codeplex.com`.

8. ACKNOWLEDGMENTS

Parts of this work have been financed by ÖNB grant no. 13.000 of the Austrian National Bank Jubilee Foundation. The research has been conducted in collaboration with Smarter Ecommerce GmbH (`http://smarter-ecommerce.com/`). Furthermore, the authors would like to thank Vitalabo Handels GmbH (`http://www.vitalabo.at/`) for providing the

Figure 9: Sensitivity analysis for recommender recall for closed sequential rules with multidimensional antecedents and consequents using lift

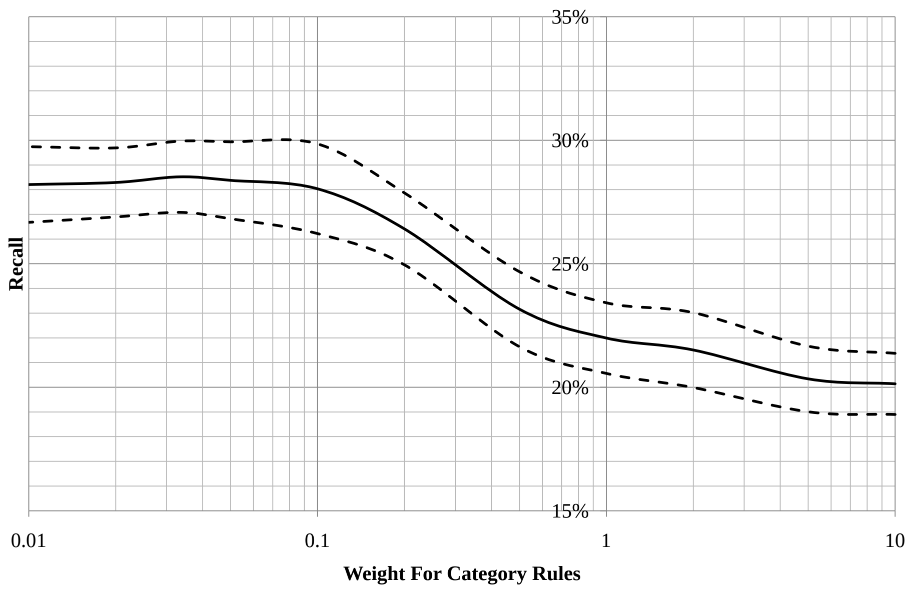
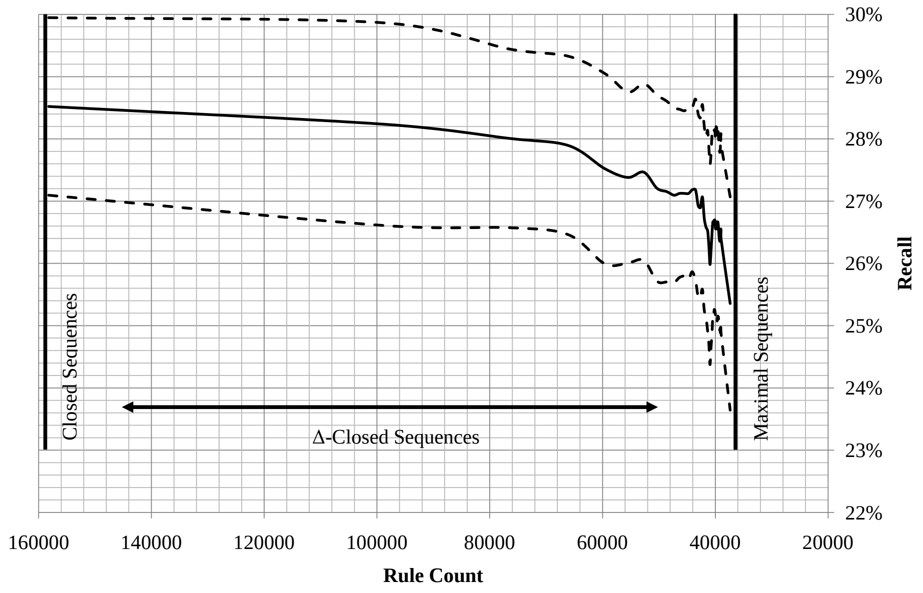


Figure 10: Sensitivity analysis for rule count for sequential rules with multidimensional antecedents and consequents using lift and a $\frac{1}{30}$ weighting for category consequents



anonymized weblog used for this evaluation.

9. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[2] B. Berendt and M. Spiliopoulou. Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB Journal*, 9:56–75, 2000. 10.1007/s007780050083.

[3] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264, New York, NY, USA, 1997. ACM.

[4] A. G. Büchner and M. D. Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Rec.*, 27(4):54–61, 1998.

[5] M. Eirinaki and M. Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Technol.*, 3(1):1–27, 2003.

[6] K. Gouda, M. Hassaan, and M. J. Zaki. Prism: A primal-encoding approach for frequent sequence mining. *Data Mining, IEEE International Conference on*, 0:487–492, 2007.

[7] K. Gouda and M. J. Zaki. Efficiently mining maximal frequent itemsets. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 163–170, Washington, DC, USA, 2001. IEEE Computer Society.

[8] Y. Hu and R. Han. An improved algorithm for mining maximal frequent patterns. In *JCAI '09: Proceedings of the 2009 International Joint Conference on Artificial Intelligence*, pages 746–749, Washington, DC, USA, 2009. IEEE Computer Society.

[9] L. T. Jonathan Herlocker, Joseph Konstan and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[10] R. Kohavi. Mining e-commerce data: the good, the bad, and the ugly. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 8–13, New York, NY, USA, 2001. ACM.

[11] M. Koufaris. Applying the technology acceptance model and flow theory to online consumer behavior. *Information Systems Research*, 13(2):205 – 223, 2002.

[12] M. Koufaris, A. Kambil, and P. A. Labarbera. Consumer behavior in web-based commerce: An empirical study. *International Journal of Electronic Commerce*, 6(2):115–138, 2001-2002.

[13] W. Lin and S. A. Alvarez. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery*, 6:83–105, 2002.

[14] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, 2000.

[15] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Discovery and evaluation of aggregate usage profiles for web personalization. *Data Mining and Knowledge Discovery*, 6:61–82, 2002. 10.1023/A:1013232803866.

[16] F. Nakagaito, T. Ozaki, and T. Ohkawa. Discovery of quantitative sequential patterns from event sequences. *Data Mining Workshops, International Conference on*, 0:31–36, 2009.

[17] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16:1424–1440, 2004.

[18] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 81–88, New York, NY, USA, 2001. ACM.

[19] M. Plantevit, A. Laurent, and M. Teisseire. Hype: mining hierarchical sequential patterns. In *DOLAP '06: Proceedings of the 9th ACM international workshop on Data warehousing and OLAP*, pages 19–26, New York, NY, USA, 2006. ACM.

[20] M. Spiliopoulou. Web usage mining for web site evaluation. *Commun. ACM*, 43(8):127–134, 2000.

[21] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT '96: Proceedings of the 5th International Conference on Extending Database Technology*, pages 3–17, London, UK, 1996. Springer-Verlag.

[22] P. Tzvetkov, X. Yan, and J. Han. Tsp: Mining top-k closed sequential patterns. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDMŠ03*, pages 347–354. IEEE Press, 2003.

[23] J. Wang, J. Han, and C. Li. Frequent closed sequence mining without candidate maintenance. *IEEE Trans. on Knowl. and Data Eng.*, 19(8):1042–1056, 2007.

[24] A. Webb. *Statistical Pattern Recognition.* John Wiley and Sons, Ltd, 2003.

[25] B. Xiao and I. Benbasat. E-commerce product recommendation agents: use, characteristics and impact. *MIS Quarterly*, 31(1):137–209, 2007.

[26] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large datasets. In *In SDM*, pages 166–177, 2003.

[27] M. J. Zaki and C. jui Hsiao. Charm: An efficient algorithm for closed itemset mining. pages 457–473, 2002.