Sequential Exceptional Pattern Discovery Using Pattern-Growth: An Extensible Framework for Interpretable Machine Learning on Sequential Data

Dennis Mollenhauer[1] and Martin Atzmueller[2]

[1] University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany
dennis@mollenhauer.is

[2] Tilburg University, Warandelaan 2, 5037 AB Tilburg, The Netherlands
m.atzmuller@uvt.nl

Abstract. Interpretable machine learning on complex data requires adequate costumizable as well as scalable computational analysis methods. This paper presents a framework combining the paradigms of exceptional model mining with sequential pattern mining in order to mine interesting patterns from complex data. We present a method for sequential exceptional pattern discovery using pattern-growth and further show how to adapt this method for large-scale complex data, with an adaptation to Map/Reduce. First evaluation results demonstrate the efficacy of the proposed method, for both synthetic as well as real-world datasets.

Keywords: Exceptional Model Mining · Sequential Patterns · Complex Data.

1 Introduction

Complex data such as large-scale multi-relational temporal and/or spatial data requires specific interpretable and explainable machine learning and analytics methods. Here the goal is to enable *human insight* into the methods and/or models in order to ultimately allow for *computational sensemaking* [6]. In particular, rule-based and pattern-based methods, e. g., [22, 23, 37] have shown considerable impact and promise in this respect.

In this paper, we consider interpretable machine learning on complex data, in particular focusing on sequence data – utilizing sequential pattern mining, which provides usually simple to interpret models similar to rule-based and association-based models, cf. [23, 37, 45]. We present the extension of sequential pattern discovery via exceptional model mining [19] – a flexible technique for systematically obtaining interesting patterns ranked by a given quality function (also called an interestingness measure), as a variant of subgroup discovery [5]. We show how to combine exceptional model mining and sequential pattern mining adapting a conceptually flexible pattern-growth approach [30, 39, 40] in order to allow for accessible customization and extension. For this, we build on the GP-Growth approach [30], while extending it for sequential pattern discovery using techniques of PrefixSpan [39, 40]. For enabling a scalable approach on large datasets, we provide an adaptation of the presented method using the Map/Reduce framework for Big Data processing. We perform first evaluations of the proposed method on synthetic as well as real-world datasets. Our preliminary results demonstrate the efficacy of the proposed method for large-scale complex data.

Our contributions are summarized as follows.

1. We propose an extension of sequential pattern mining, enabling more advanced quality functions than a simple support measure by adapting the paradigm of exceptional model mining. We present an integrated framework to discover according sequential patterns using a pattern-growth approach. With this, interpretable machine learning using exceptional model mining of sequential patterns is enabled.
2. We present an algorithm for *Sequential Exceptional Pattern discovery using Pattern-growth (SEPP)*, as the combination of exceptional and sequential pattern mining, also towards large-scale data building on the Map/Reduce framework.
3. Evaluting the proposed method, we present first results using synthetic as well as real-world data demonstrating the efficacy of the proposed method.

The rest of the paper is structured as follows: Section 2 introduces the necessary background and discusses related work. After that, Section 3 introduces the proposed method. Next, Section 4 presents our results. Finally, Section 5 concludes with a summary and interesting directions for future research.

2 Background

Recently, the concept of interpretable and explainable machine learning has received a strong interest and momentum in the data mining and machine learning community, e. g., [13, 31, 41]. In particular, in the scope complex models, in particular including black box methods, further explanation and interpretation are usually required to enable domain experts and users to understand, trust, and ultimately transform novel and useful model results into a real-world application [10, 32, 33, 42, 44]. Then, human-in-the-loop approaches [28], and computational sensemaking [6] are enabled.

In the past, rule-based/associative methods and approaches, e. g., [7, 8, 22, 23] have shown considerable impact and promising perspectives regarding interpretability and explainability, also regarding pattern mining based approaches. Here, we focus on mining sequential patterns – specifically exceptional sequential patterns – which we define according to the framework of exceptional model mining.

Below, we introduce the necessary background and discuss related work regarding these issues, focusing on (sequential) pattern mining, exceptional model mining, and the Map/Reduce framework for processing large datasets, which form the basis of our proposed methods that are introduced in the next section.

2.1 Exceptional Model Mining

Pattern mining typically aims to discover *local models* characterizing (or describing) parts of the data given an quality function, e. g., [29]. This can be achieved e. g., techniques like methods for association rule mining [3] or subgroup discovery, e. g., [5, 46]. The interestingness of the respective subgroups is usually defined by a certain property of interesting formalized by a quality function. Here, *exceptional model mining* [5, 19] can be seen as a variant of subgroup discovery, focusing on more complex quality functions, i. e., considering complex *target models*, like comparing regression models or graph structures [9]. Essentially, exceptional model mining tries to identify interesting patterns with respect to a local model derived from a set of attributes, cf. [17, 18].

2.2 Basics of Subgroup Discovery and Exceptional Model Mining

In the scope of subgroup discovery and exceptional model mining, *selectors* or *selection expressions* select the instances contained in a database D, which are covered by the respective patterns. Typical selection expression are given by attribute-value pairs in the case of nominal attributes, or by intervals in the case of numeric attributes. More formally, a *subgroup description* (or *pattern*) combines selectors into a boolean formula. For a typical conjunctive description language, a pattern $p = \{sel_1, \ldots, sel_k\}$ is defined by a set of selectors sel_j, which are interpreted as a conjunction, i.e. $p = sel_1 \wedge \ldots \wedge sel_k$. In the general case, each selector sel_k considers restrictions on the domain of an attribute contained in database D. In this paper – in our context of sequential pattern mining – we will focus on simple description languages made up of items (and itemsets), so that the selectors s_k correspond to specific items i_k or sets of those. Please note, that without loss of generality we can always construct a mapping of selectors to a set of items, if the set of selectors making up the description language is fixed. In the following, let database D contain instances with (sequential) itemset information. A *subgroup* corresponding to a pattern then contains all instances $d \in D$ for which the respective formula for the pattern evaluates to true.

For exceptional model mining, a model consists of a specific *model class* (e. g., a regression or graph-structured model, cf. e. g., [9, 17, 18]), requiring a specific quality function. It applies *model parameters* which depend on the values of the model attributes in the instances of the respective subgroup. The attributes consist of two (usually non-overlapping) sets of describing attributes and model attributes. The goal of exceptional model mining is then to identify patterns, using a specific quality function, for which the model parameters differ significantly from the parameters of the model built from the entire dataset. For a more detailed discussion we refer to [19].

Prominent approaches for exhaustive subgroup discovery and exceptional model mining, e. g., the SD-Map [12]/SD-Map* [11] and GP-Growth [30] algorithms, extend the FP-growth [26] algorithm – an efficient approach for frequent pattern mining – relying on a special data structure, the so-called FP-tree. The FP-tree is a prefix-tree built using the given set of selectors. It enables efficient access for generating frequent patterns, while containing the complete condensed frequency information for database D. Due to the limited space, we refer to [26] for more details. The basic FP-tree focuses only on frequencies, for computing the support. However, extended FP-trees, e. g., for subgroup discovery [11, 12] also contain additional information for estimating the qualities of patterns given a specific quality function. This principle is generalized in the GP-Growth algorithm [30], which substitutes simple frequencies by a generic condensed representation. This is called a *valuation basis* [30]. Essentially, it captures all the necessary (local) information to enable the computation of the applied quality function. As outlined in [30], a valuation basis is thus just represented by an n-tuple, containing all the relevant information. For computing supports of patterns, for example, just basic frequency information needs to be stored (e. g., in a 1-tuple). Then, for adapting FP-Growth to GP-Growth, the respective GP-tree structure considers valuation bases instead of simple frequencies as in the FP-tree case. Due to the limited space, we refer to [30] for a detailed discussion. Below, we outline how to apply the ideas of GP-Growth to exceptional sequential pattern mining and its efficient large-scale implementation.

2.3 Sequential Pattern Mining

Sequential pattern mining, e. g., [20, 35, 36, 39, 40, 49] aims to identify frequent subsequences in sequential databases, taking into account the order of the respective items. In the following, we define basic concepts following the formalism and notation of [40]. Let $I = \{i_1, i_2, ..., i_n\}$ denote a set of items (e. g., visited websites/locations etc., articles bought one after another, etc.). Then, a sequence $s = \langle t_1, t_2, ..., t_l \rangle$ is an ordered list (n-tuple) of n non-empty itemsets $t_i \subseteq I, |t_i| > 0, i = 1 \dots l$. An itemset t_i is represented as follows $(x_1, x_2, ..., x_m)$, with $x_i \in I$; for an itemset of size 1 we omit the brackets for simpler notation. Please note, that an item can occur several times within a sequence, but only once within an itemset. The length of a sequence is then the number of itemsets that appear in the sequence. A sequence of length l is called an l-sequence. We call a sequence $\alpha = \langle a_1, a_2, ...a_n \rangle$ a subsequence of β if it is part of a sequence $\beta = \langle b_1, b_2, ..., b_m \rangle$, i. e., $\alpha \sqsubseteq \beta$, iff there are integers $1 < j_1 < j_2 < ... < j_n \leq m$, so that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, ..., a_n \subseteq b_{j_n}$. β is then called a super sequence of α.

A sequence database S is a set of tuples (sid, s), where sid is the (unique) sequence ID and s is the sequence. If $\alpha \sqsubseteq s$ for a tuple (sid, s), then α is said to be contained in (sid, s). The frequency of a sequence α, i. e., the support in a sequence database S is determined by the number of tuples containing α. If S is a sequence database and α is a sequence, then the support of α is determined by: $support_S(\alpha) = |\{(sid, s) | (sid, s) \in S \wedge \alpha \sqsubseteq s\}|$. A sequential pattern is then defined as a sequence α with a support greater or equal than a defined minimum support threshold ξ, i. e., $support_S(\alpha) \geq \xi$, cf. [40]. The problem of sequential pattern mining is then defined as follows: Given a sequence database S and a minimal support threshold ξ, find all sequential patterns in the given database [40]. In our proposed approach, we combine sequential pattern mining with exceptional model mining enabling interpretable machine learning on sequential data.

2.4 Parallel Sequential Pattern Mining

Sequential pattern mining is a prominent data mining task, e. g., [20, 35]. Regarding large-scale data processing frameworks and parallel processing, algorithmic adaptations have been investigated, cf. [24] for a survey. One popular framework for implementing such algorithms is the Map/Reduce [16] framework. In general, Map/Reduce is applicable for a certain computational task, if this task can be divided into *independent* computational tasks, such that there is no required communication between these. Then, large tasks can be split up into subtasks – a typical divide-and-conquer strategy.

Due to these reasons, we design our proposed method for exceptional parallel sequential pattern mining using Map/Reduce - here, we can adapt the ideas of the GP-Growth algorithm using the respective valuation bases for exceptional model mining. As a method for sequential pattern mining, we build on the PrefixSpan algorithm [39, 40], due to its conceptual adaptability and similarity to the GP-Growth algorithm in partitioning the data. With this, PrefixSpan can then be parallelized with Map/Reduce as we will describe below in more detail. Essentially, we can simply extend the support counter of the algorithm by a valuation basis. Further constraints and extensions can then be simply implemented using both the PrefixSpan as well as the Exceptional Model Mining techniques. With this, we provide an extensible pattern-based framework for interpretable machine learning on sequential data.

3 Method

For extending sequential pattern mining to exceptional sequential pattern mining, one important step is to identify a suitable sequential pattern mining algorithm. In this paper, we extend the PrefixSpan algorithm since it is "compatible" with the GP-Growth approach w.r.t. our proposed extension for exceptional model mining and the incorporation of valuation bases, in particular the aggregation operation. From [30] we recall the definition of a valuation basis, following the notation and terminology proposed in [30]:

Definition 1. *For constructing valuation bases on an arbitrary set V, we consider an abelian semi-group $(V, \oplus)$, where $\oplus$ is a binary operator on V, such that $\oplus : V \times V \rightarrow V$, and the following holds: (1) V is closed under $\oplus$, i. e., $a, b \in V \Rightarrow a \oplus b \in V$, (2) $\oplus$ is associative, i. e., $a, b, c \in V \Rightarrow a \oplus (b \oplus c) = (a \oplus b) \oplus c$, and (3) $\oplus$ is commutative, i. e., $a, b \in V \Rightarrow a \oplus b = b \oplus a$. An element $v \in V$ is then called a valuation basis.*

The support (count) is a very simple example of a valuation basis (and its domain) which can be simply aggregated. We refer to [30] for more details on more complex valuation bases, specifically for more complex model classes such as the variance or the linear regression model. Basically, the variance model capturing the variance of a numeric model attribute in a sequential pattern can be represented using a 3-tuple, from which the variance (aggregation) can be computed; similarly, the aggregation can be computed for the linear regression model (for two model attributes) using a 5-tuple.

For pattern-growth algorithms, the PrefixSpan algorithm is a good candidate since it is conceptually flexible and *extensible* including various variants, e. g., [4, 15, 25, 43] such that e. g., domain-driven constraints and extensions can be easily implemented and integrated into a customized approach. Below, we first describe the problem of sequential exceptional model mining, briefly summarize the PrefixSpan algorithm, before we present the approaches for sequential exceptionality detection using PrefixSpan.

3.1 Sequential Exceptional Model Mining

Below, we first define the problem of *Sequential Exceptional Model Mining (SEMM)*. Essentially, with *SEMM* the problems of exceptional model mining and sequential pattern mining are combined. Sequential exceptional patterns are those for which their model has a deviation in comparison to the overall model of the database from which they were extracted. Let $e = (eid, ea, s)$ denote an extended sequence, where eid is a unique identifier of e, ea are the model attributes for estimating the model parameters and s is a sequence. Let $E = \{e_1, e_2, ..., e_i\}$ be a database of extended sequences.

Definition 2. *We consider an extended sequence database, a model M, a minimum support threshold ξ, a quality function q_M and an integer k. Sequential exceptional model mining is the task of finding the k best sequential patterns w.r.t. the quality function q_M, with a support greater or equal to the minimum support threshold ξ.*

It is easy to see, that if ξ is set to one, then all possible patterns according to the given quality function are retrieved. The threshold ξ was introduced as a separate criterion in order to provide a way to limit individual outliers and restrict the problem size easily. However, it could potentially also be integrated into the quality function directly.

Basically, the respective requirements for the sequential pattern mining algorithm follow from the requirements of Map/Reduce and those of the valuation bases. The Map/Reduce framework requires that each dataset (sequence) can be viewed individually and that the data can be partitioned. This is obviously given for the sequential pattern mining problem, since each sub-sequence can be counted individually, and partitions can be formed at the borders of sequences. Furthermore, valuation bases only guarantee a $\oplus$ operator, which can be understood as an addition. Since the support counter is to be extended by valuation bases, it is necessary that the algorithm always adds the support.

The algorithm PrefixSpan [39, 40] by Pei et. al. works with partial databases, each of which is a projection of the entire sequence database with respect to one prefix. In the comparative tests in [35], PrefixSpan performed well throughout and was the only algorithm that was able to perform all test runs.

Therefore, since PrefixSpan is a conceptually flexible yet effective algorithm which can be extended as already discussed above, and can be parallelized on Map/Reduce, we utilize this algorithm for our approach. In the following, we describe the PrefixSpan algorithm. We assume that all elements of a sequence are lexicographically ordered. A sequence $\beta = \langle e_1, e_2, ..., e_n \rangle$ is a prefix of a sequence $\alpha = \langle e'_1, e'_2, ..., e'_m \rangle$, iff (1) $e'_i = e_i$ for $(i \leq m-1)$, (2) $e'_m \subseteq e_m$, and (3) all items in $(e_m - e'_m)$ occur lexicographically after those in e'_m. For example, $\langle (ad)c \rangle$, $\langle (ad) \rangle$ and $\langle a \rangle$ are prefixes of $\langle (ad)c(bc)(ae) \rangle$, but not $\langle ac \rangle$ nor $\langle (ad)b \rangle$.

Let $\alpha = \langle e_1, e_2, ..., e_n \rangle$ denote a sequence and $\beta = \langle e_1, e_2, ...e_{m-1}, e'_m \rangle$ a prefix of α. Then $\gamma = \langle e''_m, e_{m+1} \rangle$ is a suffix, where $e''_m = (e_m - e'_m)$. The connection of prefix β with suffix γ to a sequence α is also represented as $\alpha = \beta \cdot \gamma$.

Algorithm 1 PrefixSpan: outputs the complete set of sequential patterns

procedure PREFIXSPAN(α, l, $S|_\alpha$)
- ▹ α: prefix (on the first pass *null*)
- ▹ l: length of α
- ▹ $S|_\alpha$: DB projected after α, or the complete database during the first run

$B \leftarrow$ search $S|_\alpha$ and find frequent items b so that:
1. b can be added to the last itemset of α to form a sequential pattern, or
2. $\langle b \rangle$ can be added to α as a new itemset to form a sequential pattern

for all b in B **do**
- $\alpha' = \alpha \cdot b$
- $support \leftarrow support_S|_a$(b)
- STDOUT(α' + $support$)
- $S|_{\alpha'} \leftarrow$ project $S|_\alpha$ to b
- PREFIXSPAN(α', $l + 1$, $S|_{\alpha'}$)

In Algorithm 1, we denote the basic PrefixSpan algorithm. PrefixSpan expects a sequence database and a minimal support threshold as parameters. PrefixSpan assumes, without loss of generality, that the itemsets of the sequences are in a total order, where usually a lexicographic order is used. During the initial PrefixSpan call, frequent 1-sequences are determined. Afterwards the passed sequence database is projected to

each frequent item and PrefixSpan is called recursively with the respective 1-sequence as prefix. In the first recursion stage, the transferred database is searched for frequent 2-itemsets with respect to the prefix. Then for each 2-sequence found, the database is projected and PrefixSpan is called with the respective 2-sequence as the prefix. These recursive calls end when there are no more frequent n-sequences in the respective projected database or the respective projected databases are empty.

The central idea of PrefixSpan is to split the database into projections – an idea which is similar to the conditional FP-trees of FP-Growth. A projected database is defined as follows [40]:

Definition 3. *Let α be a sequential pattern in a sequence database S. Then the α projected database $S|_\alpha$ will contain all the α suffixes from the database S.*

The support in the projected databases is determined as follows:

Definition 4. *Let α be a Sequential Pattern in a sequence database S and β be a sequence with prefix α. Then the support of β in the α projected database $S|_\alpha$ is the number of sequences γ in $S|_\alpha$, so that $\beta \sqsubseteq \alpha \cdot \gamma$ applies [40].*

The database $S|_\alpha$ constructed in this way cannot be larger than S. This database contains all the necessary suffixes to find all sequential patterns starting with the prefix α. For a more efficient implementation, the use of pseudo sequences is suggested. When projecting a sequence, then no new sequence is created, but the original sequence is referenced and only a pointer is used to point to the location where the projection begins. A detailed discussion can be found in [40].

In the algorithm, counting the support is only performed when determining the frequent items. At this point the valuation basis can be calculated as follows. A test must be performed on each sequence in the transferred database to determine whether the sequence can be extended. If this is the case, the counter is increased by one for the corresponding pattern. Assuming that the sequence is extended by a valuation basis, i.e. $s = \langle sid, valuationBasis, sequence \rangle$, the additional *valuationBasis* attribute can be used at this point to determine the resulting valuation basis, which is then stored in the result. Therefore, during determining the support, all valuation bases for the respective sequential patterns are available and can be evaluated. This is how the according valuation bases for a specific exceptional model class can be integrated into PrefixSpan, if the according model class can be represented in this way. For more details on limitations and restrictions we refer to [30] for a detailed discussion.

3.2 Sequential Exceptional Pattern Discovery using Pattern-Growth (SEPP)

Due to its flexibility and strict partitioning of the problem, PrefixSpan is very well suited for adaptation and for being parallelized using Map/Reduce. For our pattern-growth approach, we adapt and extend PrefixSpan, as shown in Algorithm 2. The determination of frequent sequences has not been changed. Then, the extension using valuation bases discussed above is straight-forward, by adding valuation bases to each sequence as additional information. With these valuation bases, the quality of an exceptional sequential pattern can then be simply estimated using the respective quality function. We call this general approach *sequential exceptional pattern discovery using pattern-growh (SEPP)*.

We call our method for parallelizing PrefixSpan using Map/Reduce *parallel SEPP*, while we can also resort to a serial version for smaller problem sizes. The basic idea of *parallel SEPP* is to convert the projection and the determination of the frequent sequences into a parallel formulation. Since it is problematic to implement recursive algorithms with Map/Reduce, *(parallel) SEPP* we convert the depth-first search strategy of PrefixSpan into a breadth-first search strategy. By replacing the call stack with a queue only the order in which the projections are visited changes, cf. [38]. This order is not important for PrefixSpan, because there is no relationship between projections of the same level. Each recursive call holds at most one projection of a prefix with the length $n+1$, if its predecessor has the length n. The frequent patterns of length n are then searched for in parallel and then the projections are calculated in parallel. This process is repeated until no more new patterns are found or all projections are empty. Initially, only frequent 1 sequences are searched. (Parallel) SEPP is depicted in Algorithm 2.

Algorithm 2 SEPP: outputs the complete set of sequential exceptional patterns

procedure SEPP(S, ξ)
▷ S: a sequence database
▷ ξ: minimal support
n-sequences ← find all common 1 sequences in parallel
STDOUT(n-sequences)
projections ← new set
for all α in n sequences **do** ▷ parallel
projections ← projections + PROJECT(S to α)
while projections != blank **do**
n-sequences ← search parallel every α-projection
and find frequent items b, so that either:
1. b can be added to the last itemset of α to form a sequential exceptional pattern,
2. $\langle b \rangle$ can be added to α as a new itemset to form a sequential exceptional pattern
STDOUT(n-sequences)
projectionsNew ← new set
for all $S|_\alpha$ in projections **do** ▷ parallel
α'-sequences ← any sequence in n-sequences matching $S|_\alpha$
for all α' in α'sequences **do**
projectionsNew ← projectionsNew + PROJECT($S|_\alpha$ to α')
projections ← projectionsNew

(Parallel) SEPP does not change the projections and the way frequent sequences are determined. It simply modifies the processing order so that counting steps for length n and projections of length n can be executed in parallel. Each counting and projection step is performed as one Map/Reduce job. For example, twice as many Map/Reduce jobs are required to execute parallel SEPP as the maximum pattern length. If we only take into account the support of a sequential pattern, then parallel SEPP simplifies to a parallelized version of a sequential pattern mining algorithm. Since longer patterns result in a lot of small projections, from a certain size on a projection can be processed in a serial way, i. e., not applying parallelization using Map/Reduce – as *serial SEPP*.

3.3 Interpretable Machine Learning on Sequential Data using SEPP

Figure 1 depicts the proposed framework for interpretable machine learning on sequential data – using our proposed combination of exceptional model mining and sequential pattern mining using SEPP. It includes the workflow from data, to extended sequences (contained in a database), to sequential patterns and finally a pattern-based model. Two further important components are an *exceptionality model* capturing the exceptional model information, and a *human-in-the-loop* approach enabled by the interpretable SEPP method.

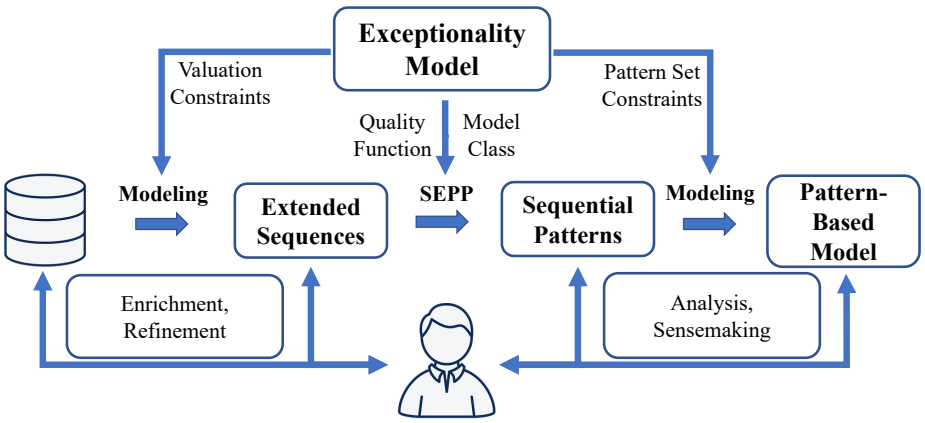

Fig. 1. Interpretable ML Framework – Sequential Exceptional Pattern Discovery.

In a human-centered approach, pattern-based modeling is performed in an iterative approach, which can also result in incremental modeling. Both the exceptionality model as well as the human-in-the-loop provide important information and constraints for the process, e. g., by including specific (objective) design constraints for the valuation basis, or by providing (subjective) domain-driven interestingness constraints. Basically, the process starts with the modeling of the extended sequence database given the original data. Here, valuation constraints, e. g., for building and modeling valuation bases and domains can be applied, ultimately resulting in the extended sequence database. These, can also be enhanced using refinement and enrichment steps by the human-in-the-loop. Next, the SEPP algorithm is applied – given the respective quality function for the applied model class. Here, depending on the problem size, either the serial or the parallel algorithm is applied. Finally, the obtained patterns are provided to the pattern-based model, which can also incorporate pattern set constraints given the exceptionality model. Both of these enable analysis and sensemaking capabilities for the human-in-the-loop, relying on the applied inherently interpretable pattern-based approach [22, 23, 37].

4 Results

For evaluation, we created a synthetic dataset using the IBM Quest synthetic data generator [2]. In addition, we utilized two real-world datasets, as described below.

For the experiments, the Hadoop cluster of the IT Service center of the University of Kassel was used. This cluster consisted of twelve nodes. Each node had two AMD Dual-Core Opteron 2218 processors (2.6 GHz), with 16 GB of RAM and about 960 GB free disk space. The Hitachi Ultrastar A7K2000 hard drives were connected with SATA. The nodes were running a Red Hat Enterprise Linux Server 6.2 with Apache Hadoop 1.1.1 running on Orcale Java 1.7. The cluster had two map and two reduce slots configured on each node, providing a total of 24 mappers and reducers.

4.1 Synthetic Dataset

The IBM Quest Synthetic Data Generator [2] was used to generate the synthetic data, as a standard data generator in such evaluation settings, e. g., [35, 39, 40, 47, 48]. The Quest generator provides synthetic shopping cart transactions, simulating real-world shopping basket composition. Here, most sequences have a similar length and only a few are particularly long. The same applies to the number of items per transaction.

For the evaluation, we created a dataset with 27.8 million sequences using the parameters shown in Table 1. This dataset will be referenced as *27M* in the following. The number of itemsets per sequence is determined using a Poisson distribution, the parameter μ is equal to $|C|$. The same procedure is used to determine the size of the itemset, here $\mu = |T|$. Furthermore, the frequency of the items is exponentially distributed [1].

Table 1. Parameters of the IBM Quest Synthetic Data Generator.

$\|D\|$	Number of customers (= number of sequences)
$\|C\| = 8$	Average number of itemsets per customer
$\|T\| = 2.5$	Average number of items per itemset
$\|S\| = 4$	Average length of sequences
$\|I\| = 1.25$	Average size of itemsets
$N_S = 5000$	Number of sequences
$N_I = 25000$	Number of itemsets
$N = 20000$	Number of different items

4.2 Parallel SEPP with a Simple Exceptional Model Class (Support)

First the runtime was examined regarding different support thresholds with 24 mappers and reducers (Figure 2), where we applied a simple exceptional model class and quality function (support – as the standard baseline for sequential pattern mining). The runtime investigation in [40] showed with similarly structured datasets that the runtime of PrefixSpan approximately doubles when the support threshold is divided by two. Figure 2

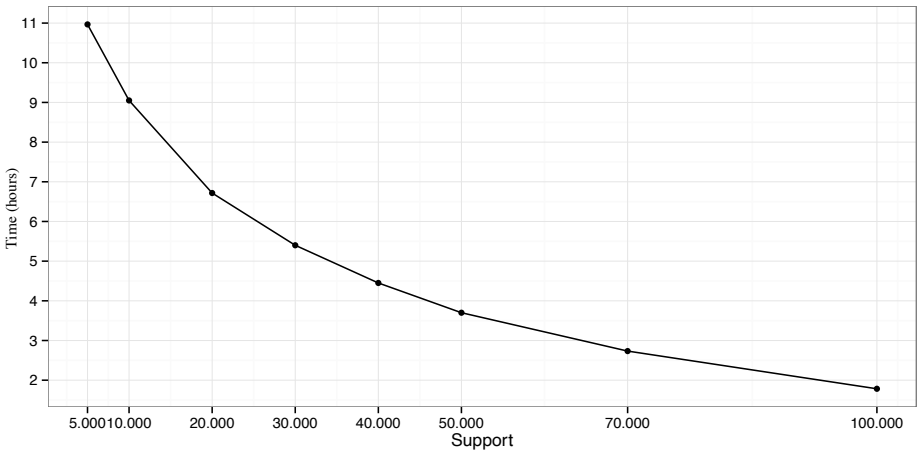

Fig. 2. SEPP (support quality function), 27M: Duration in relation to support.

shows that SEPP provides similar or better results. For a minimum support between 100000 and 50000 the factor is 2, between 20000 and 10000 the factor is only 1.3.

In a further series of tests the speedup of SEPP was examined. Speedup specifies the factor of acceleration of a parallel program when the number of processors is increased.

Definition 5. *The speedup S_p is defined as $S_p = \frac{T_1}{T_p}$, where p represents the number of processors, T_1 the runtime with one processor, and T_p the runtime with p processors.*

For an ideal speedup $S_p = p$. This ideal result is usually difficult to achieve due to the computational overhead involved in managing the respective computational tasks. In order to be able to run this experiment in an acceptable time, a minimum support of 100000 was chosen. With this support the calculation with 24 mappers and reducers took about two hours. Assuming that an almost ideal speedup is achieved, a runtime of two days can be expected with one mapper and reducer. The result of this test can be seen in Figure 3. We observe, that for up to 24 reducers the speedup is close to the ideal speedup. Since only 24 slots are available for mapper/reducers, it was expected that no further increase could be observed with 48 mappers and reducers, as also shown in the experiments. So, for the evaluation the algorithm fulfilled all expectations above, demonstrating its scalability and efficacy for processing and analyzing complex datasets. In particular, with the 27M dataset, it could be shown that SEPP scales almost linearly for larger datasets and can calculate even larger datasets with very small minimum support thresholds in less than twelve hours.

4.3 Parallel SEPP With More Complex Exceptional Model Classes

For more complex model classes bases than support – as discussed above – the quality estimation implemented in SEPP is basically an extension of the standard support calculation. Therefore, for more complex valuation bases only a constant factor determines

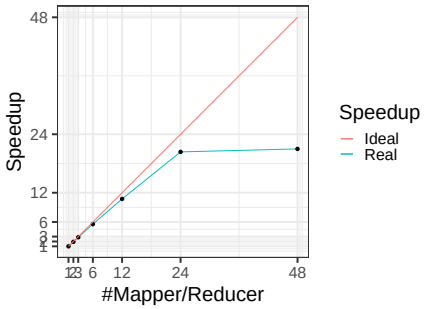

Fig. 3. Speedup of SEPP (support quality function) on 27M, in relation of the applied number of Mappers and Reducers.

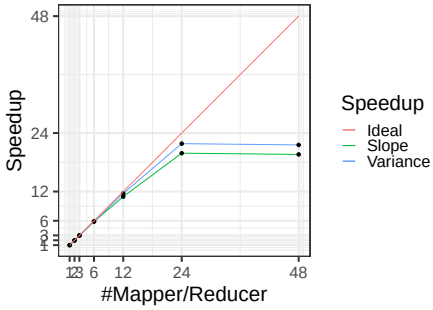

Fig. 4. Corresponding speedup of SEPP on 27M, using the simple linear regression model (Slope) and variance model (Variance).

the runtime of SEPP compared to only using support. To confirm this, the experiments shown in Figure 2 were repeated with more complex valuation bases. As a valuation input, the number of items per sequence was calculated in advance. As valuation domain the variance model was used. Furthermore, we applied the simple linear regression model, where we took the first two items of a sequence as a valuation input - i. e., as the two respective variables (independent, dependent) for the linear regression modeling.

The results of these experiments are shown in Table 2, where the column "Factor" indicates the ratio of the runtime of the variance model divided by runtime using simple support. This factor is about 1.5 for all support thresholds. We observed a similar behavior for the linear regression model class.

Table 2. SEPP runtimes of variance model and support model classes in minutes on dataset 27M.

Support	**Variance Model Class**	**Support Model Class**	**Factor**
100000	157	107	1,47
70000	239	164	1,46
50000	327	222	1,47
30000	490	324	1,51
10000	803	543	1,48

In a further series of experiments, the speedup for the variance model and the simple linear regression model was determined. The result is shown in Figure 4. The nearly linear speedup of parallel SEPP is achieved with different model classes. However, for the simple linear regression model the speedup is slightly worse than for the variance model. This can be explained by the fact that the variance model's valuation base is only a 3-tuple, whereas the simple linear regression model uses a 5-tuple base. This leads to a higher effort when sorting and copying the data between mappers and reducers.

4.4 Results on Real-World Datasets

Two real-world datasets were applied for evaluation, i. e., a dataset from MSNBC, and data from a web server protocol of a Hungarian news portal. For our experiments, we applied a simple valuation basis for SEPP, i. e., using the support as quality function.

MSNBC Dataset The dataset "msnbc.com anonymous web data" from the UCI Machine Learning Repository [21,27] is a dataset containing a formatted web server log of the site msnbc.com for 28.09.1999. The dataset contains all requests for the entire day that were not answered by caches. The calls were divided into 17 thematic categories, such as main page, news, technology, local. A sequence of these categories was created for each user. In total the dataset contains 989818 sequences with an average number of 5.7 calls per user. Each category contains between ten and 5000 requests [27].

Table 3. Runtimes MSNBC with a support of 7500 and a limit for serial processing of 10000.

Reducer	48	24	12	6	3	2	1
Terminal (min)	20	16	15	18	18	22	30

Table 3 shows the results of seven runs with different numbers of reducers. The parameter minimum support was set to 7500 and the limit from which size the projected databases should be processed with the serial SEPP variant (serial limit) was set to 10000 sequences. Since this dataset is small in relation to the synthetic dataset, only a very low speedup can be observed. With a larger number of reducers even the runtime increased. When running three reducers, the step of the sequential PrefixSpan dominated the calculation with nine minutes. In the calculation with twelve reducers this step took only 6.5 minutes. In both cases it could be observed that all but one reducer were finished within two minutes and only the last one had to be waited for. This suggests that there are relatively dense partitions with less than 10000 sequences in the data or that the distribution of instances between the reducers is not balanced.

In order to confirm that on this data set no significant improvement of the runtime with more than twelve mappers and reducers can be achieved, several runs with low support thresholds were performed without serial processing. The result is shown in Figure 5. As expected, the optimum is given for twelve mappers and reducers.

Table 4. Lead times Kosarak data set; serial limit 10000.

Support	**Reducer**	48	24	12	6	3	2	1
7500	**Terminal (min)**	16	13	13	14	19	23	38
5000	**running time (min)**	18	16	16	18	24	30	50
2500	**Terminal (min)**	29	27	29	33	49	59	NA

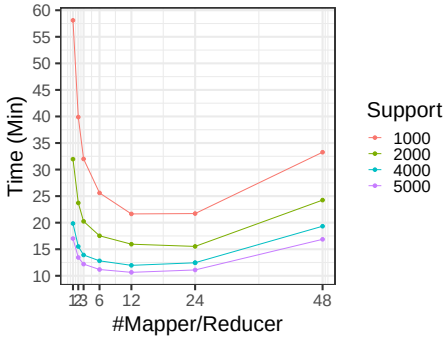

Fig. 5. Runtime of SEPP on the MSNBC dataset with different minimum support thresholds, in relation to different numbers of mapper/reducers.

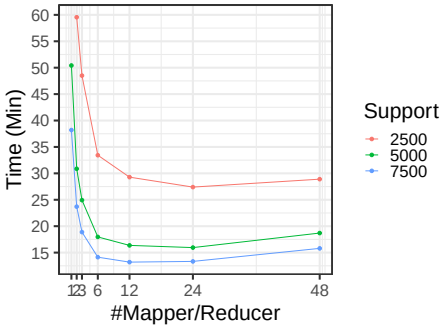

Fig. 6. Runtime of SEPP on the Kosarak dataset with different minimum support thresholds, in relation to different numbers of mapper/reducers.

Kosarak Dataset The second applied real-world dataset[3] considers a Hungarian news portal and contains anonymous access sequences of 990002 users. The sequences cover 41270 different pages. The average sequence length is 8.09, the longest 2498.

We applied minimum supports of 5000, 7500 and 2500, with a serial limit of 10000. The runtimes are shown in Table 4 and Figure 6. The optimal number of mappers and reducers in the first two cases was twelve reducers, as for the MSNBC dataset (see Section 4.4). For a support of 2500 the optimum was 24 reducers. As for the synthetic data with only 24 available slots for mappers/reducers, it was expected that no further increase could be observed with 48 mappers/reducers, as also shown in the experiments.

5 Conclusions

In this paper, we have proposed a pattern-based extensible framework for interpretable machine learning on complex data, in particular focusing on sequence data – utilizing sequential pattern mining. We showed how to combine exceptional model mining and sequential pattern mining into an extensible approach, building on existing sequential pattern mining methods, in particular pattern-growth approaches. We presented the according method for sequential exceptional pattern discovery using pattern-grown (SEPP). For this basic algorithm, both serial as well as parallelized variants were presented. We evaluated the proposed method on synthetic as well as real-world datasets. Our preliminary results demonstrated the efficacy of the proposed methods.

For future work, we aim investigate more complex quality functions as well as interpretability constraints, which we aim to integrate into the proposed approach, also in different application contexts. In addition, we aim to compare the proposed framework to further parallelized implementations, e. g., [24]. Yet another interesting direction for future work is to investigate extended interpretability and sensemaking patterns during the application of the proposed approach, e. g., [14, 34].

[3] FIMI Repository: http://fimi.uantwerpen.be/

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. Tech. rep., Research Report RJ 9910, Almaden Research Center, IBM Research Division, San Jose, California (Oktober 1994)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proc. ICDE. pp. 3 –14. IEEE (1995)
3. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. VLDB. pp. 487–499. Morgan Kaufmann (1994)
4. Antunes, C., Oliveira, A.L.: Generalization of pattern-growth methods for sequential pattern mining with gap constraints. In: Proc. MLDM. pp. 239–251. Springer (2003)
5. Atzmueller, M.: Subgroup Discovery. WIREs DMKD **5**(1), 35–49 (2015)
6. Atzmueller, M.: Declarative Aspects in Explicative Data Mining for Computational Sensemaking. In: Proc. International Conference on Declarative Programming. Springer (2018)
7. Atzmueller, M., Baumeister, J., Puppe, F.: Quality Measures and Semi-Automatic Mining of Diagnostic Rule Bases. In: Proc. INAP/WLP. pp. 65–78. No. 3392 in LNAI, Springer (2005)
8. Atzmueller, M., Baumeister, J., Puppe, F.: Semi-Automatic Learning of Simple Diagnostic Scores Utilizing Complexity Measures. Artif. Intell. Med. **37**(1), 19–30 (2006)
9. Atzmueller, M., Doerfel, S., Mitzlaff, F.: Description-Oriented Community Detection using Exhaustive Subgroup Discovery. Information Sciences **329**, 965–984 (2016)
10. Atzmueller, M., Hayat, N., Schmidt, A., Klöpper, B.: Explanation-Aware Feature Selection using Symbolic Time Series Abstraction: Approaches and Experiences in a Petro-Chemical Production Context. In: Proc. IEEE INDIN. IEEE Press, Boston, MA, USA (2017)
11. Atzmueller, M., Lemmerich, F.: Fast Subgroup Discovery for Continuous Target Concepts. In: Proc. ISMIS. LNCS, vol. 5722, pp. 1–15. Springer, Berlin, Germany (2009)
12. Atzmueller, M., Puppe, F.: SD-Map - A Fast Algorithm for Exhaustive Subgroup Discovery. In: Proc. PKDD. pp. 6–17. Springer, Berlin, Germany (2006)
13. Biran, O., Cotton, C.: Explanation and Justification in Machine Learning: A Survey. In: IJCAI-17 Workshop on Explainable AI (2017)
14. Bloemheuvel, S., Kloepper, B., van den Hoogen, J., Atzmueller, M.: Enhancing sequential pattern mining explainability with markov chain probabilities (abstract). In: DBDBD (2019)
15. Chaudhari, M., Mehta, C.: Extension of prefix span approach with grc constraints for sequential pattern mining. In: Proc. ICEEOT. pp. 2496–2498. IEEE (2016)
16. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (Jan 2008)
17. Duivesteijn, W., Knobbe, A., Feelders, A., van Leeuwen, M.: Subgroup Discovery Meets Bayesian Networks–An Exceptional Model Mining Approach. In: Proc. ICDM. IEEE (2010)
18. Duivesteijn, W., Feelders, A., Knobbe, A.J.: Different Slopes for Different Folks: Mining for Exceptional Regression Models with Cook's Distance. In: Proc. KDD. pp. 868–876 (2012)
19. Duivesteijn, W., Feelders, A.J., Knobbe, A.: Exceptional Model Mining. Data Mining and Knowledge Discovery **30**(1), 47–98 (2016)
20. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. Data Science and Pattern Recognition **1**(1), 54–77 (2017)
21. Frank, A., Asuncion, A.: UCI mach. learning repository. http://archive.ics.uci.edu/ml (2010)
22. Fürnkranz, J., Kliegr, T.: A brief overview of rule learning. In: International symposium on rules and rule markup languages for the semantic web. pp. 54–69. Springer (2015)
23. Fürnkranz, J., Kliegr, T., Paulheim, H.: On cognitive preferences and the plausibility of rule-based models. Machine Learning **109**(4), 853–898 (2020)
24. Gan, W., Lin, J.C.W., Fournier-Viger, P., Chao, H.C., Yu, P.S.: A survey of parallel sequential pattern mining. ACM Transactions on Knowledge Discovery from Data **13**(3), 1–34 (2019)
25. Han, J., Pei, J., Yan, X.: Sequential pattern mining by pattern-growth: Principles and extensions. In: Foundations and advances in data mining, pp. 183–220. Springer (2005)

26. Han, J., Pei, J., Yin, Y.: Mining Frequent Patterns Without Candidate Generation. In: Proc. ACM SIGMOD Intl. Conference on Management of Data. pp. 1–12. ACM Press (05 2000)
27. Heckerman, D.: msnbc.com anonymous web data. http://archive.ics.uci.edu/ml/machine-learning-databases/msnbc-mld/msnbc.data.html (1999)
28. Holzinger, A.: Interactive machine learning for health informatics: when do we need the human-in-the-loop? Brain Informatics **3**(2), 119–131 (2016)
29. Knobbe, A.J., Cremilleux, B., Fürnkranz, J., Scholz, M.: From Local Patterns to Global Models: The LeGo Approach to Data Mining. In: Proc. ECML/PKDD LeGo Workshop (2008)
30. Lemmerich, F., Becker, M., Atzmueller, M.: Generic Pattern Trees for Exhaustive Exceptional Model Mining. In: Proc. ECML/PKDD. Springer, Berlin, Germany (2012)
31. Li, X., Huan, J.: Constructivism Learning: A Learning Paradigm for Transparent Predictive Analytics. In: Proc. SIGKDD. pp. 285–294. ACM (2017)
32. Liu, S., Wang, X., Liu, M., Zhu, J.: Towards Better Analysis of Machine Learning Models: A Visual Analytics Perspective. Visual Informatics **1**(1), 48–56 (2017)
33. Lonjarret, C., Robardet, C., Plantevit, M., Auburtin, R., Atzmueller, M.: Why should I trust this item? Explaining the recommendations of any model. In: Proc. IEEE International Conference on Data Science and Advanced Analytics. pp. 526–535. IEEE (2020)
34. Lycett, M., Marshan, A.: Capturing sensemaking pattern during data analysis: A conceptual framework. In: Proc. ISD 2016. Springer, Berlin/Heidelberg, Germany (2016)
35. Mabroukeh, N.R., Ezeife, C.I.: A taxonomy of sequential pattern mining algorithms. ACM Computing Surveys **43**(1), 3:1–3:41 (Dezember 2010)
36. Mathonat, R., Nurbakova, D., Boulicaut, J.F., Kaytoue, M.: Seqscout: Using a bandit model to discover interesting subgroups in labeled sequences. In: Proc. IEEE International Conference on Data Science and Advanced Analytics. pp. 81–90. IEEE (2019)
37. Molnar, C.: Interpretable Machine Learning. Lulu. com (2020)
38. Ottmann, T., Widmayer, P.: Algorithmen und Datenstrukturen. Spektrum, Heidelberg (2012)
39. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns by prefix-projected growth. In: Young, D.C. (ed.) Proc. International Conference on Data Engineering. pp. 215–224. IEEE, Los Alamitos (April 2001)
40. Pei, J., Han, J., Mortazavi-Asl, B., W., J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Transactions on Knowledge and Data Engineering **16**(11), 1424 – 1440 (November 2004)
41. Ribeiro, M.T., Singh, S., Guestrin, C.: Why Should I Trust You?: Explaining the Predictions of Any Classifier. In: Proc. ACM SIGKDD. pp. 1135–1144. ACM (2016)
42. Samek, W., Wiegand, T., Müller, K.R.: Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. arXiv preprint arXiv:1708.08296 (2017)
43. Shabtay, L., Yaari, R., Dattner, I.: A guided fp-growth algorithm for multitude-targeted mining of big data. arXiv preprint arXiv:1803.06632 (2018)
44. Sternberg, E., Atzmueller, M.: Knowledge-Based Mining of Exceptional Patterns in Logistics Data: Approaches and Experiences in an Industry 4.0 Context. In: Proc. International Symposium on Methodologies for Intelligent Systems. LNCS, vol. 5722. Springer (2018)
45. Vojíř, S., Zeman, V., Kuchař, J., Kliegr, T.: Easyminer. eu: Web framework for interpretable machine learning based on rules and frequent itemsets. Knowl.-Based Syst. **150** (2018)
46. Wrobel, S.: An Algorithm for Multi-Relational Discovery of Subgroups. In: Proc. PKDD. pp. 78–87. Springer (1997)
47. Zaki, M.J.: Efficient enumeration of frequent sequences. In: Gardarin, G., French, J.C., Pissinou, N., Makki, K., Bouganim, L. (eds.) Proc. CIKM. pp. 68–75. ACM, New York (1998)
48. Zaki, M.J.: Parallel sequence mining on shared-memory machines. Journal of Parallel and Distributed Computing **61**(3), 401 – 426 (2001). https://doi.org/10.1006/jpdc.2000.1695
49. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine learning **42**(1-2), 31–60 (2001)