Noname manuscript No.
(will be inserted by the editor)

Methods for explaining Top-N recommendations through subgroup discovery

Mouloud Iferroudjene · Corentin Lonjarret · Céline Robardet · Marc Plantevit · Martin Atzmueller

the date of receipt and acceptance should be inserted later

Abstract Explainable Artificial Intelligence (XAI) has received a lot of attention over the past decade, with the proposal of many methods explaining black box classifiers such as neural networks. Despite the ubiquity of recommender systems in the digital world, only few researchers have attempted to explain their functioning, whereas one major obstacle to their use is the problem of societal acceptability and trustworthiness. Indeed, recommender systems direct user choices to a large extent and their impact is important as they give access to only a small part of the range of items (e. g., products and/or services), as the submerged part of the iceberg. Consequently, they limit access to other resources. The potentially negative effects of these systems have been pointed out as phenomena like echo chambers and winner-take-all effects, because the internal logic of these systems is to likely enclose the consumer in a "déjà vu" loop. Therefore, it is crucial to provide explanations of such recommender systems and to identify the user data that led the respective system to make the individual recommendations. This then makes it possible to evaluate recommender systems not only regarding their effectiveness (i. e., their capability to recommend an item that was actually chosen by the user), but also with respect to the diversity, relevance and timeliness of the active data used for the recommendation. In this paper, we propose a deep analysis of two state-of-the-art models learnt on four datasets based on the identification of the items or the sequences of items actively used by the models. Our proposed methods are based on subgroup discovery with different pattern languages (i. e.,

Mouloud Iferroudjene
École nationale Supérieure d'Informatique (ESI), Algiers, Algeria.
E-mail: gm_iferroudjene@esi.dz

Corentin Lonjarret · Céline Robardet
Univ. Lyon, INSA Lyon, CNRS, LIRIS UMR 5205, F-69621 Villeurbanne, France

Marc Plantevit
Univ. Lyon, CNRS, LIRIS UMR 5205, F-69622 Villeurbanne, France

Martin Atzmueller
Osnabrück University, Semantic Information Systems Group, Osnabrück, Germany, and
German Research Center for Artificial Intelligence (DFKI), Osnabrück, Germany
E-mail: martin.atzmueller@uni-osnabrueck.de

itemsets and sequences). Specifically, we provide interpretable explanations of the recommendations of the Top-N items, which are useful to compare different models. Ultimately, these can then be used to present simple and understandable patterns to explain the reasons behind a generated recommendation to the user.

1 Introduction

A recommender system offers personalized recommendations that are based on the history of users in the system and their respective resemblance to the histories of other users. Personalizing a suggestion then consists of filtering the content in order to only keep the most relevant items for a given user. Since such processes are typically opaque, being able to explain a recommendation should increase the user's confidence and trust in the system [Pu and Chen, 2006]. This question has already been considered, for classification problems (e. g., [Ribeiro et al., 2016]), but also in the scope of recommender systems (e. g., [Tintarev and Masthoff, 2011]) to some extent. However, these approaches did not extend to a deep analysis of complex recommendation models nor to methods providing model-agnostic explanations. This is the task that we tackle in this paper, restricting our analysis to recommender systems that only use the user's history in the system (i. e., the users' sequences), without taking into account external information, e. g., profile data or item content of a user. Furthermore, we focus on Top-N recommendations instead of considering the best (Top-1) recommendation only.

Recommender systems are used to retrieve information and products that are most relevant to a user. Existing techniques and methods are numerous and – even if we can understand them from a theoretical point of view – it is usually difficult to understand a particular recommendation. However, being able to identify the "foundations" of a recommendation – i. e., answering the "why" question using an *explanation* – would help to increase the confidence that one can have in it [Tintarev and Masthoff, 2011], and also to assess and compare different methods that may only differ slightly. It is generally accepted that a recommendation can be based on a "global" characterization of the user, which generally relates to their "user preference" in the literature. It can also depend on recent user activity, and/or sequential relations between a set of items (like watching episodes of a series in order). In our context, we call this "sequential dynamics".

Thus, we propose a method for explaining recommender systems with the objective to answer the following questions, which we address in this paper:

1. Which actions in the past are behind the recommendation?
2. Is the (sequential) order of the actions important for the recommendation?
3. In the case that the order of actions is important, which sequences of actions do support the recommendation?
4. Furthermore, can we interpret a model globally by identifying a typology of possible recommendations? That is to say, can we identify a small number of explanations that represent the whole model?
5. Finally, how can we compare the "explanations" of several models?

In preliminary work [Lonjarret et al., 2020], we proposed two methods to explain the first item recommended by a recommender system. In this paper, we present a non-trivial extension of this work towards the explanation of the Top-N

recommendations. Indeed, recommender systems generally offer an ordered list of items to the user who does not only consider the first item, but a set of items classified at the top by the system. In particular, compared to [Lonjarret et al., 2020], we significantly extended the method to take into account and explain the Top-N recommendations. This necessitated studying a new pattern language to support the explanations and led us to extend the experiments accordingly.

The proposed method (see Figure 1 for an overview) exploits the neighborhood of the user's sequence to study the variability of the Top-N recommendation made by the model:

1. Given a user sequence u, it generates sequences of items close to u. Two different neighborhoods $US_1(u)$ and $US_2(u)$ are considered:
   (a) $US_1(u)$ consists of all subsets of the items contained in the user sequence, where the items are ordered as in the user sequence. With this neighborhood, the goal is to identify items that strongly impact the recommendation.
   (b) $US_2(u)$ focuses on the impact of the item order on the recommendation. Thus, $US_2(u)$ consists of perturbed sequences, resulting from the permutation of two consecutive items in the user sequence – a random number of times (i. e., given u, an arbitrary number of permutations is applied).
2. These sequences, close to the original one, are used to identify the key items on which the recommendation is based. First, the recommender model $\mathcal{R}$ is used to compute the recommendations scores obtained on the new sequences. For analysis and for uncovering relevant patterns explaining the recommendation, we apply subgroup discovery [Wrobel, 1997, Atzmueller, 2015] to identify the past actions that play an important role in the recommendation of the Top-N items TN^u. Subgroup discovery is part of the family of rule-based models which are widely accepted as the most interpretable ones [Fürnkranz et al., 2020]. As subgroup discovery is defined to characterize a single target variable, we extend this method for the explanation of TN^u items by constructing an aggregate value of their associated scores provided by the recommender system that is in turn used to guide the discovery of subgroups having an exceptionally high score on this target. We then have two options:
   (a) On the sequences of US_1, our proposed subgroup discovery approach identifies user actions that lead the model to rank the Top-N items obtained when considering the sequence u also in first positions of the recommendation. To that end, we use an itemset description language and apply the SD-Map algorithm [Atzmueller and Puppe, 2006, Lemmerich et al., 2016].
   (b) For the sequences US_2, we first check whether the order between past actions matters via a simple distributional test. We use sequences as the description language and the algorithm SEPP [Mollenhauer and Atzmueller, 2020] for sequential analysis in order to identify the sequential patterns that lead to the Top-N recommendations in high positions.
3. For both, the best subgroup allows us to identify conditions on the user's actions maximizing a target variable aggregating the positions of the items TN^u in the recommended sequences.

Our contributions are summarized as follows: (1) We present an approach for explaining recommendations of a determined model, but which can be based on any type of recommender systems (what is called model-agnostic), exploiting the recommendation history. The proposed methodological approach for generating specific sets of recommendation histories is independent of the model type and

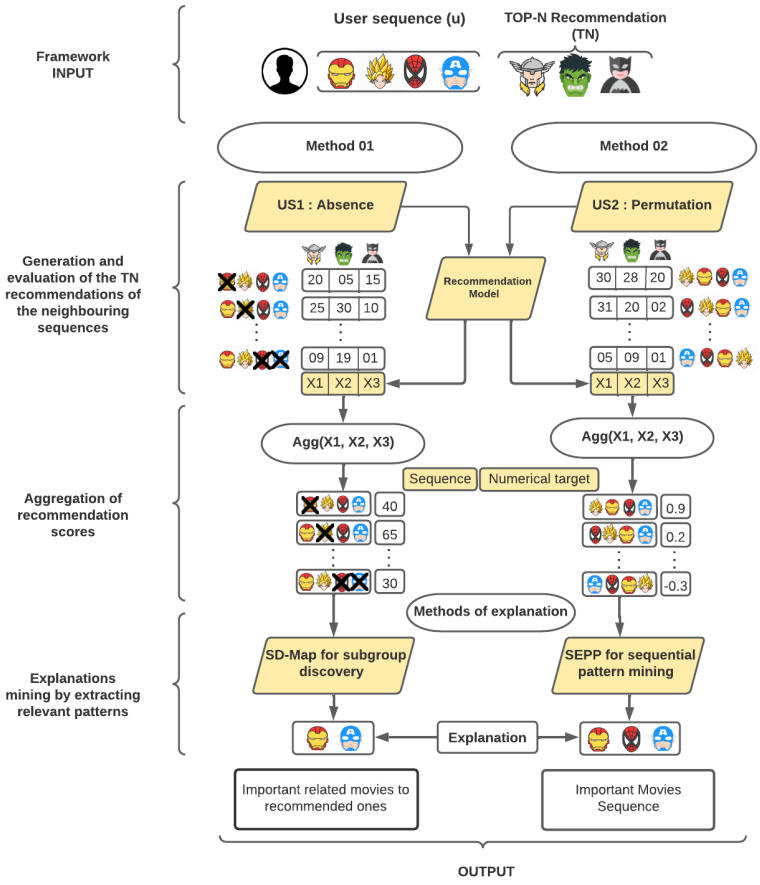


Fig. 1: Overview of how the explanation of Top-N recommendations is created given a user sequence and the corresponding recommended items TN^u.

thus broadly applicable. (2) Instantiating this approach, we present an explanation method utilizing subgroup discovery [Wrobel, 1997, Atzmueller, 2015] for identifying active data used for recommendation. (3) Finally, we present experiments performing a deep analysis of two state-of-the-art models – known to perform well on Top-N sequential recommendation – learned on four datasets. We provide an extensive discussion of our results in context.

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

2 Related Work and Background

Below, we first discuss general related work on model explanation and interpretation, particularly including model-agnostic and post-hoc explanation methods. Next, we move on to explanations on recommender systems.

2.1 Model Explanation

Recently, the concept of transparent and explainable models has gained a strong focus and momentum in the data mining and machine learning community. While the methods sketched above focus on a specific machine learning model type, there are several approaches for model-agnostic explanation methods (i.e., methods independent of the model type), e. g., [Ribeiro et al., 2016,Ribeiro et al., 2018]. A first family of such methods is based on the production of counterfactual explanations, e. g., [Mandel, 2007], seeking to identify a part of an instance at the origin of the prediction. A second family includes techniques based on data perturbation and randomization for the study of black box models, e. g., [Henelius et al., 2014]. The approach we propose in the following also considers perturbation techniques, but we specialize them in the particular case of recommender systems. Specifically, we apply subgroup discovery methods to provide post-hoc explanations that provide the following advantages: (1) The approach can be applied to any black box recommendation model – making it model-agnostic; (2) subgroup discovery is an important method for obtaining descriptive patterns (rules) which are rated as one of the most interpretable types of patterns [Fürnkranz et al., 2020].

2.2 On Explanation in Recommender Systems

Explanation and *explanation-aware* approaches have been widely investigated in different disciplines, e. g., in artificial intelligence, data science, etc. e. g., [Schank, 1986, Wick and Thompson, 1992, Roth-Berghofer et al., 2007]. In [Roth-Berghofer and Cassens, 2005], Roth-Berghofer and Cassens outline the combination of goals and kinds of explanations, in the context of case-based reasoning. Sørmo et al. [Sørmo et al., 2005] suggest a set of explanation goals addressing transparency, justification, relevance, conceptualisation, and learning. Explanation goals specifically help to focus on user needs and expectations towards explanations. They aim at addressing to understand what and when the system has to be able to explain (something). For recommender systems, different approaches for providing explanations have been studied, e. g., [McSherry, 2005, Tintarev and Masthoff, 2011] targeting mainly content-based, collaborative filtering, and case-based approaches since explanation-awareness is an important factor for supporting the user, e. g., [Tintarev and Masthoff, 2011]. Here, explanation is mainly integrated into the respective method. In contrast, for (black box) machine learning methods to be used for recommendation, explanations have been largely neglected. Our paper investigates and tackles this problem, proposing a framework and method for model-agnostic explanations only utilizing the recommendation history.

2.3 Subgroup Discovery

Subgroup discovery [Klösgen, 1996, Wrobel, 1997, Atzmueller, 2015] has been established as a general and broadly applicable technique for descriptive and exploratory data mining. In general, it aims at identifying subgroups, i. e., of individuals that are *interesting* with respect to a given quality function, e. g., estimating the difference in shares of a binary target concept in the subgroup vs. the overall dataset. For a binary target concept, for example, we are then interested in large subgroups with a high share of individuals for which the target concept is true.

In general, a *database* $D = (R, A)$ is given by a set of data records R, also called instances, and a set of attributes A. For nominal attributes, a *basic pattern* $(a_i = v_j)$ is a Boolean function $R \rightarrow \{0, 1\}$ that is true if the value of attribute $a_i \in A$ is equal to v_j for the respective data record. The set of all basic patterns is denoted by Σ. A *subgroup description* or (complex) *pattern* P is given by a set of basic patterns $P = \{p_1, \ldots, p_l\}, p_i \in \Sigma, i = 1, \ldots, l$, interpreted as a conjunction $p_1 \wedge \ldots \wedge p_l$, with $length(P) = l$. A pattern can thus also be interpreted as the *body* of a *rule*. The rule *head* then depends on the property of interest, e. g., for a binary target concept T on a basic pattern $sel_T = true$. A *subgroup* $S_P := ext(P) := \{r \in R | P(r) = true\}$, is the set of all data records that are covered by the subgroup description P.

In the case of a database containing items/itemsets, we can directly map our (general) database D to an item database I, where each data record $r \in R$ contains a set of items $I_r \subseteq 2^I$. That is, the attribute–value pairs discussed above are represented as binary variables $A_i, i = 1 \ldots m, m = |I|$ which represent the respective items contained in I.

Subgroups are computed using a specifiable interestingness measure $q\colon 2^{\Sigma} \rightarrow \mathbb{R}$, that makes it possible to retrieve the k best subgroups [Atzmueller, 2015]. For a binary target concept, let n be the size of $ext(P)$, the subgroup described by the pattern P (i. e., its *support*) and the share t_P of the target concept in the subgroup, i. e., its *confidence*, are combined by the interestingess measures q_S as follows: $q_S(P) = n \cdot (t_P - t_0)$, where t_0 denotes the (default) share of the target concept in the database D, or by the *Lift* quality function $q_L(P) = \frac{t_P}{t_0}$.

2.4 Sequential Pattern Mining

Sequential pattern mining, e. g., [Fournier-Viger et al., 2017, Pei et al., 2004, Mabroukeh and Ezeife, 2010] aims to identify frequent subsequences in sequential databases, taking into account the order of the respective items. In general, 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 [Pei et al., 2004]. There are various algorithms for sequential pattern mining, e. g., [Zaki, 2001, Pei et al., 2001, Pei et al., 2004, Mathonat et al., 2019] while the extension to subgroup discovery has been proposed in [Mollenhauer and Atzmueller, 2020] in the form of the SEPP algorithm. For our approach, we apply SEPP for mining exceptional sequential patterns in the form of subgroups. As discussed in Section 3, we consider an itemset database, taking the subgroup description language $\mathcal{S}$ made of all sequences of items of I without repetition: $\mathcal{S} = \mathit{\Pi}_{k=0}^{|I|} 2^{|I|-k}$.

We propose a new perturbation-based technique for recommender system explanation that borrows from subgroup discovery on itemset and sequential databases. As we seek to explain the first N items recommended by a system, we need to extend subgroup methods to explain multiple target values for sequential patterns and itemsets. We detail how we do this below.

3 Explanations for Top-N recommendation

We propose a model-agnostic approach for local explanation in the context of recommender systems. We consider a specific user and analyze the recommendations made according to their user history (i. e., the previous interactions within the system on which the recommendation is based). Similar to [Ribeiro et al., 2016], we explore the neighborhood of these recommendations. Specifically, our objective is to isolate the items from the user's history which lead to the recommendations. In our previous work [Lonjarret et al., 2020], we used subgroup discovery to identify the active data related to the Top-1 recommendation (the first item recommended to the user). Indeed, subgroup discovery makes it possible to identify the relationships between such a *target variable* (or target attribute) and some *explaining variables*. The proposed method consists of first generating profiles close to that of the studied user and calculating the recommendations for these profiles using the recommender system. Second, the set of Top-1 recommendations is analyzed in order to isolate the groups or sequences of items for which the item initially recommended exhibits a high rank using subgroup discovery.

However, the quality of a recommendation is generally not evaluated solely on the basis of the Top-1 item, but on a larger set, generally ordered, whose size is sufficiently reduced to be considered by the user. We call this ordered set the Top-N recommendation. Hence, we propose to extend the Top-1 explanation method based on subgroup discovery to the Top-N explanations. The challenge here is to make subgroup discovery work when there are several target values and also to consider sequential pattern languages for which very recent methods were proposed. The overview of the method is presented below.

3.1 Overview of the method

Considering a user sequence u and its associated Top-N recommendation TN^u, we propose to apply subgroup discovery [Wrobel, 1997] to identify the *active data* used for the recommendation – essentially, to identify the relationships between the Top-N recommendation and the items of the user history that lead to the recommendation, i. e., a subset of u in our context. We are interested in two types of explanations E_u: the items that "trigger" the recommendations of the Top-N items, and the order of those items that impact the recommendation. Therefore, we consider two different subgroup description languages to identify the items and their order that are related to the recommendation. The subgroup discovery process consists of generating a set US of neighboring sequences of u, to apply the recommendation model on these sequences ($\mathcal{R}(s) = t^s, s \in US$), and then to find subgroups of sequences associated to a description d of the considered language

$\mathcal{R}$	The recommender system to be studied.
U	The set of user sequences.
I	The items to be recommended.
N	The number of items to be explained.
u	The user sequence used by the recommender system to generate the sequence TN^u ($\mathcal{R}(u) = TN^u$).
$TN^u = \langle i_1^\star, \dots i_N^\star \rangle$	The Top-N items recommended by the recommender system. They are to be explained.
$GT^u = \langle i_1, \dots i_m \rangle$	The ground truth items for the user u, last 20% of the user's history.
s	Any sequence.
I^s	The set of items that appear in s.
$r^s_{i^\star}$	The score obtained by item $i^\star$ when a sequence s is given to the recommender system.
t^s	the Top-N items returned by the model when considering the sequence s
E_u	An explanation found for the sequence u.
$t^{u \setminus E_u}$	The Top-N recommended items when considering the sequence u from which the items of the explanation E_u (defined for the sequence u) have been removed.
$US_1(u)$	Neighborhood sequences of u made of sequences by removing one or several items from u.
$US_2(u)$	Neighborhood sequences of u made of sequences where only the items order of u changes.
$\mathcal{I}$	Subgroup description language for SD-Map defined as all possible sets of items from I.
$\mathcal{S}$	Subgroup description language for SEPP made of all sequences of items of I without repetition.

Table 1: Summary of notations used throughout the paper.

such that the Top-N recommendations on these sequences are as close as possible to the one of sequence u (see the toy example in Table 2).

	$u = \langle 1, 2, 3, 4, 5, 6 \rangle$	$\mathcal{R}(u) = \langle 7, 8 \rangle = TN^u$
US	$s_1 = \langle 1, 2, 3, 4, 5 \rangle$	$\mathcal{R}(s_1) = \langle 7, 10 \rangle$
	$s_2 = \langle 1, \underline{2, 3, 5, 6} \rangle$	$\mathcal{R}(s_2) = \langle 7, 8 \rangle$
	$s_3 = \langle 1, 4, 5 \rangle$	$\mathcal{R}(s_3) = \langle 6, 5 \rangle$
	$s_4 = \langle 1, 2, 4, 5 \rangle$	$\mathcal{R}(s_4) = \langle 10, 8 \rangle$
	$s_5 = \langle \underline{2, 3}, 4, \underline{5, 6} \rangle$	$\mathcal{R}(s_5) = \langle 7, 8 \rangle$

Table 2: Example of the method: we are looking for a description d which, when present in the data, leads to a recommendation similar to that of u. For example, the sequential description $d = \langle 2, 3, 5, 6 \rangle$ covers sequences s_2 and s_5 whose lead to the same recommendation as u.

Our subgroup discovery approach is used for identifying the *explaining variables* d associated to a large value on a *target variable*, which we model using a function Agg that summarizes the position of the TN^u items in the recommendation made for a sequence s. It looks for the description d that has a large value regarding the *qmean* measure (inspired by the simple binomial quality function for numerical

target attributes, c. f., [Lemmerich et al., 2016]), defined as

$$qmean(d, TN^u, US) = \sqrt{|SG(d)|}\left(\frac{\sum_{s\in SG(d)} Agg(\mathcal{R}(s), TN^u)}{|SG(d)|} - \frac{\sum_{s\in US} Agg(\mathcal{R}(s), TN^u)}{|US|}\right) \quad (1)$$

where $SG(d)$ is the subset of sequences of US that satisfy the description d. Large $qmean$ values indicate that the function Agg is larger on average on the subgroup than on the whole set of sequences.

In the following sections, we detail (1) the way neighboring sequences are generated, (2) the function Agg which we use, (3) the description languages which we consider, and finally (4) the algorithms for computing the subgroups.

3.2 Neighborhood generation

To identify active items used for recommendation we explore two neighborhoods given a user sequence u:

1. To focus on the impact of a group of items on the recommendation, we generate neighboring sequences $US_1(u)$ by removing one or several items from u. The possible number of such sequences is equal to $2^{|u|}$. If the recommendation is mainly based on a group of items, then removing these items from the sequences would have a great impact on the recommendation.
2. To assess the importance of the item order on the user sequence u, we consider another set of neighboring sequences $US_2(u)$ consisting of sequences where only the order changes. Starting with the observed sequence u, we shuffle it by randomly picking an item of the sequence and swapping it with one of its neighbors. This process is repeated n times where n is chosen randomly in the interval $[5, 25]$. Thus, the parameter n is used to control the amount of disturbance incurred by the sequence. At the end of this process, the perturbed sequence is added to the neighborhood. We produce K such perturbed sequences to form the neighborhood[1].

Depending on the applied description language (as discussed below) we use one or the other neighborhood.

3.3 Target modeling: Agg functions used to summarize the set of the Top-N items

Let us consider a Top-N recommendation made of N items, $TN^u = \{i_1^\star, \dots i_N^\star\}$ based on a user sequence u. Let $r_{i^\star}^s$ be the score obtained by item $i^\star$ when a sequence s is given to the recommender system. In the following, we consider four functions that can be used as function Agg in Equation (1).

[1] NB: we removed duplicated perturbed sequences in a post-processing step.

3.3.1 SumScore: Maximizing the sum of the scores

This function considers the sum of the scores returned by the recommender system for each item:

$$SumScore(s, TN^u) = \sum_{i^\star \in TN^u} r^s_{i^\star} \tag{2}$$

These values are highly dependent on the recommender system used and can not be used to compare results from different models.

3.3.2 Kendall: Kendall τ score

To measure how the order of the items is conserved between the two sequences and also to check if the items of TN^u are not too far apart from each other in the recommendation, we use the Kendall τ measure [Kendall, 1938] to count how many pairs of items are concordant or discordant between the recommendations based on sequence u and the one resulting from sequence s:

$$P(s, TN^u) = |\{(k, \ell) \mid i^\star_k, i^\star_\ell \in TN^u \text{ and } r^u_{i^\star_k} > r^u_{i^\star_\ell} \text{ and } r^s_{i^\star_k} > r^s_{i^\star_\ell}\}|$$

$$Q(s, TN^u) = |\{(k, \ell) \mid i^\star_k, i^\star_\ell \in TN^u \text{ and } r^u_{i^\star_k} > r^u_{i^\star_\ell} \text{ and } r^s_{i^\star_k} < r^s_{i^\star_\ell}\}|$$

where $P(s, TN^u)$ is the number of concordant pairs of items of TN^u between sequences u and s and $Q(s, TN^u)$ is the number of discordant pairs. The Kendall measure is thus obtained by:

$$Kendall(s, TN^u) = \frac{P(s, TN^u) - Q(s, TN^u)}{\frac{[TN^u|(|TN^u|-1)}{2}} \tag{3}$$

where the denominator is given by the total number of possible pairs, so the values of τ range between -1 and 1.

3.3.3 KScore: Penalizing SumScore when s and TN are not correlated

SumScore is a measure that does not take into account the order of items returned by the recommender system. To correct this aspect we can combine the two previous measures into a single score:

$$KScore(s, TN^u) = Kendall(s, TN^u) \times SumScore(s, TN^u) \tag{4}$$

Thus, the closer the order of s is to the order of the recommendation TN, the greater *KScore*. If the two orders are not correlated, the score tends towards 0 and if they are anti-correlated, then the score takes a negative value.

3.3.4 KSBounded: normalizing SumScore and Kendall before combining them

To allow comparisons between recommender systems (and thus to overcome the ranges of score values returned by each model), we normalize the *SumScore* and *Kendall* values before their combination. For both measures, we use a min-max normalization:

$$SumScoreMinMax(s, TN^u) = \frac{SumScore(s, TN^u) - |TN^u| \times \min_{i^\star \in TN^u}(r^s_{i^\star})}{\max i^\star \in TN^u(r^s_{i^\star}) - \min i^\star \in TN^u(r^s_{i^\star})}$$

$$KendallMinMax(s, TN^u) = \frac{(Kendall(s, TN^u) + 1)}{2}$$

The final *KSBounded* score is given by:

$$KSBounded(s, TN^u) = KendallMinMax(s, TN^u) \times SumScoreMinMax(s, TN^u) \quad (5)$$

3.4 Description languages

Our subgroup discovery approach aims to characterize subsets of sequences associated with high target variable values (here: *qmean* values). We propose to use two description languages to describe the subgroups: one where the description is a set of items, the other where a subgroup is characterized by a sub-sequence of items.

3.4.1 Description language $\mathcal{I}$

The subgroup description language $\mathcal{I}$ used for active item identification is defined as all possible sets of items from I: $\mathcal{I} = 2^I$. A descriptive pattern $d \in \mathcal{I}$ *covers* a user sequence s iff d is included in I^s ($d \subseteq I^s$) the set of items that appear in s. Thus, considering a set of user sequences US, the subgroup of US described by d, i. e., the *pattern cover*, is made of all the sequences of US that contain all the items of d: $SG(d) = \{s \in US \mid d \subseteq I^s\}$.

3.4.2 Description language $\mathcal{S}$

To identify the potential order effect on the recommendations, we consider the subgroup description language $\mathcal{S}$ made of all sequences of items of I without repetition: $\mathcal{S} = \Pi_{k=0}^{|I|} 2^{|I|-k}$. A pattern $d \in \mathcal{S}$ covers a sequence s, $d \sqsubseteq s$, iff d appears in s in the same order: Let $d = \langle i_1, \ldots, i_k \rangle$ and $s = \langle s_1, \ldots, s_\ell \rangle$, there exists indices $j_1 < \cdots < j_k$ such that $s_{j_h} = i_h$, $\forall h = 1 \ldots k$. Considering a set of user sequences US, the subgroup of US described by d is the set of sequences of US selected by d: $SG(d) = \{s \in US \mid d \sqsubseteq s\}$.

3.5 Subgroup Discovery: Algorithms to compute subgroups of description language $\mathcal{I}$ and $\mathcal{S}$ associated with large numerical values

In this section, we briefly summarize the used algorithms for subgroup discovery for language $\mathcal{I}$, i. e., the SD-Map/SD-Map* algorithm, and for language $\mathcal{S}$, i. e., SEPP, following their respective presentation in [Atzmueller and Puppe, 2006, Lemmerich et al., 2012, Lemmerich et al., 2016, Mollenhauer and Atzmueller, 2020].

3.5.1 SD-Map/SD-Map*

Prominent approaches for exhaustive subgroup discovery, e. g., the SD-Map [Atzmueller and Puppe, 2006]/SD-Map* [Lemmerich et al., 2016] algorithms, and GP-Growth [Lemmerich et al., 2012] for subgroup discovery and exceptional model mining, extend the FP-growth [Han et al., 2000] algorithm. It enables efficient access for generating frequent patterns, while containing the complete condensed frequency information for a database. The basic FP-tree focuses only on frequencies, for computing the support. However, extended FP-trees for subgroup discovery [Atzmueller and Puppe, 2006, Lemmerich et al., 2016] also contain additional information for estimating the qualities of patterns given a specific quality function. This principle is generalized in the GP-Growth algorithm [Lemmerich et al., 2012], which substitutes simple frequencies by a generic condensed representation that captures all the necessary (local) information to enable the computation of the applied quality function. For subgroup discovery, SD-Map applies the given quality function in order to determine the top-k subgroups, i. e., those with the top-k qualities according to the given function. In our case, we apply the *qmean* quality function to find the best subgroup:

$$\text{SD-Map}(TN^u, US) = \max_{d \in \mathcal{I}} qmean(d, TN^u, US) = E^u$$

3.5.2 SEPP

The novel SEPP algorithm described in [Mollenhauer and Atzmueller, 2020] focuses on exceptional sequential pattern mining. Below, we summarize the main concepts of SEPP, following the presentation and notation of [Mollenhauer and Atzmueller, 2020]. For extending sequential pattern mining to exceptional sequential pattern mining, one important step is to identify a suitable sequential pattern mining algorithm. For the SEPP algorithm [Mollenhauer and Atzmueller, 2020], the basic idea is to extend the PrefixSpan algorithm [Pei et al., 2001, Pei et al., 2004]. Most importantly, this extension is also "compatible" with the GP-Growth approach w.r.t. the given extension for exceptional model mining, c. f., [Mollenhauer and Atzmueller, 2020]. With this technique, SEPP is able to include valuation bases, as introduced in [Lemmerich et al., 2012]. Originally, valuation bases were devised as a generalization of simple counts, e. g., as used in computing the support of a pattern. In the algorithm, valuation bases are then used for computing the quality of the respective patterns. The support (count) is a very simple example of a valuation basis (and its domain) which can be simply aggregated. Essentially, valuation bases then abstract the calculation and aggregation of values of parts of the given database for the applied quality function. We refer to [Lemmerich et al., 2012] for more details on more complex valuation bases. Essentially, SEPP

then allows combining the problems of exceptional model mining and sequential pattern mining, as described in [Mollenhauer and Atzmueller, 2020]. In particular, 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.

Following [Mollenhauer and Atzmueller, 2020], 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 (in our case of items). Let $E = \{e_1, e_2, ..., e_i\}$ be a database of extended sequences. Then, considering 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. As introduced in [Mollenhauer and Atzmueller, 2020], 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. In our case, we apply the *qmean* quality function, and suitable minimal support thresholds as detailed in the experiments:

$$\text{SEPP}(TN^u, US) = \max_{d \in \mathcal{S}} qmean(d, TN^u, US) = E^u$$

3.6 Demonstration of the process workflow on a toy example

In this section, we illustrate the functioning of our Top-N recommendations explaining method through a simple example of the execution of the different concepts presented in the previous section. We consider the user (historical) sequence u associated with a list of Top-N recommendations, denoted TN^u of length 3 ($|TN^u| = 3$). As a function for aggregating the scores Agg, we take the function *KSBounded* (defined in Equation 5) which returns normalized values, therefore easier to understand.

	US_1(s)			US_2(s)	
ID	Perturbed sequence	Agg(TN^u)	IDs	Perturbed sequences	Agg(TN^u)
1	$\langle 12, 580, 5\rangle$	**0.91**	1	$\langle 12, 580, 5\rangle$	**0.97**
2	$\langle \emptyset, \emptyset, 5\rangle$	**0.84**	2	$\langle 5, 580, 12\rangle$	0.70
3	$\langle \emptyset, 580, \emptyset\rangle$	0.24	3	$\langle 12, 5, 580\rangle$	0.32
4	$\langle \emptyset, 580, 5\rangle$	**0.88**	4	$\langle 580, 12, 5\rangle$	**0.98**
5	$\langle 12, \emptyset, \emptyset\rangle$	0.25	5	$\langle 5, 12, 580\rangle$	0.71
6	$\langle 12, \emptyset, 5\rangle$	**0.98**	6	$\langle 580, 5, 12\rangle$	0.29
7	$\langle 12, 580, \emptyset\rangle$	0.12			

Table 3: Example of $US_1(u)$ and $US_2(u)$ neighborhood generations with $u = \langle 12, 580, 5\rangle$ and TN^u=$\langle 1, 55, 89\rangle$ (values greater than 0.8 are highlighted).

We present in Table 3 the generation of $US_1(u)$ and $US_2(u)$ neighborhoods with u=$\langle 12, 580, 5\rangle$. Since there are three items in the sequence u, we can generate 7 sequences in US_1(u) (2^3-1 as we do not consider the empty sequence) and six possible

sequences for US_2(u) (the number of possible permutations is $|US_2(u)|$=$|u|$!= 3!=6). Hence, for each perturbed sequence, we compute the aggregate recommendation score of the items in TN^u: $Agg(TN^u)$.

SD-Map			**SEPP**		
ID	Explanation	qmean	IDs	Explanation	qmean
1	$\{5\}$	4,12	1	$\langle 580, 5\rangle$	2.19
2	$\{12, 5\}$	3.02	2	$\langle 12, 5\rangle$	2.19
3	$\{580, 5\}$	2,97	3	$\langle 12, 580, 5\rangle$	1.13
4	$\{12, 580, 5\}$	1.88	4	$\langle 580, 12, 5\rangle$	1.13

Table 4: Example of explanations generated by SD-Map and SEPP for $US_1(u)$ and $US_2(u)$ in Table 3.

Table 4 shows the 4 best explanations generated by SD-Map and SEPP, using US_1(u) and US_2(u) neighborhoods from Table 3. The best explanations are $\{5\}$ for SD-Map and $\langle 580, 5\rangle$ for SEPP. These results make sense and seem significant. In fact, for $US_1(u)$, we can clearly see that the best scores are obtained when item $\{5\}$ is present in the user's history. On the other hand, for $US_2(u)$, we can easily notice that the aggregate score of the items $i^\star \in TN^u$ with the *KSBounded* function are high (close to 1) when $\{5\}$ is the last item the user has consulted. Therefore $\langle 580, 5\rangle$ and $\langle 12, 5\rangle$ have the same qmean value (the quality function is described in Equation 1).

3.7 Complexity

The complexity of our methods is due to the complexity related to the neighborhood generations and the one due to the computation of the explanations on them. The prediction of the score or rank of the TN^u items by the recommender system has an important impact on the complexity of the respective neighborhood generations. Each recommender system has its own computational complexity, denoted as RS_{Comp}. Thus, computing $US_1(u)$ has a complexity in $O(RS_{Comp} \times 2^{|u|})$, while the complexity of $US_2(u)$ is in $O(RS_{Comp} \times |I \setminus u| \times K)$, where K is number of generated sequences. We limit the complexity for computing $US_1(u)$ by only considering users with a history size $|u|$ between 2 and 15. The impact of this limitation is evaluated in Section 4.4.

Considering the computation of the explanations based on $US_1(u)$, we use SD-Map [Atzmueller and Lemmerich, 2009] instantiated for numeric target attributes. As discussed in [Atzmueller and Lemmerich, 2009] the complexity of SD-Map grows exponentially with the number of items. However, with the efficient pruning approaches implemented by SD-Map [Atzmueller and Lemmerich, 2009, Lemmerich et al., 2016] the runtime is typically significantly reduced in practical applications. The explanations based on $US_2(s)$ are obtained thanks to SEPP, where similar considerations as for SD-Map apply, while the complexity is even larger due to the sequential pattern language. However, as for SD-Map, our implementation of SEPP also applies efficient (optimistic estimate) pruning,

c. f., [Lemmerich et al., 2016] for a general overview, which enables considerably reduced runtimes in practice.

4 Experiments

In this section, we evaluate our method through several experiments. We first describe the real-world datasets we consider and the experimental setup. Then, we present a quantitative study of the explanations provided by our method. We study the performance of the explanations through several metrics. Eventually, we provide some specific examples of explanations in our recommendation context. We implemented our methods in Python and performed the experiments on a machine equipped with 8 Intel(R) Xeon(R) W-2125 CPU @ 4.00GHz cores 126GB main memory, running Debian GNU/Linux. The code and data are made available[2].

4.1 Datasets, Models and Aims

Datasets: We evaluate our framework on both sparse and dense datasets from different domains by considering four different benchmarks:

- **Amazon** was introduced by [McAuley et al., 2015]. It contains Amazon product reviews from May 1996 to July 2014 from several product categories. We have chosen to use the *Video games* categories as we can easily visualize the products and have an intuition on the recommendations made.
- **MovieLens 1M**[3] [Harper and Konstan, 2015] is a popular dataset including 1 million movie ratings from 6040 users between April 2000 and February 2003. We pre-processed the dataset, selecting the most recent 50 ratings for each user.
- **Foursquare** [Falher et al., 2015] depicts a large number of user check-ins on the Foursquare website from December 2011 to April 2012.
- **Adressa** [Gulla et al., 2017] includes news articles (in Norwegian). The dataset was offered by Adresseavisen, a local newspaper company in Trondheim, Norway.

For each dataset, ratings are converted into implicit feedback, and we only consider users and items with at least five interactions. Table 5 shows the datasets' characteristics. These characteristics include the number of users (having at least five interactions), the number of items, the total number of actions, the average size of the user sequences, the average time a user takes an item, the density of the dataset, and the concentration which is the percentage of actions that are related to the Top 5% most popular items. Table 5 shows that the datasets are diverse, each having its own specificity. ML-50 exhibits the highest density while the density and the concentration are much greater for Adressa than others. The Foursquare dataset has the biggest number of users and items but the lowest density. Eventually, the Video-games dataset is rather "square", i.e., the number of user is similar to the number of items. All these characteristics witness that the benchmarks we consider are diverse. This supports thorough and systematic experimental study.

[2] Source code link.

[3] `http://grouplens.org/datasets/movielens/1m/`

Table 5: Main characteristics of the datasets after preprocessing (users and items that have at least 5 interactions): # U is the number of users, # I is the number of different items, # A is the total number of items (with duplicates), #A/#U is the average size of user sequences, #A/#I is the average time an item is taken by a user, Density is # A/(# U× # I), Concentration is the proportion of the Top 5% of the most popular interacting items.

Datasets	#U	#I	#A	#A/#U	#A/#I	Density	Concentration
Video games	31013	23715	287107	9.26	12.11	0.04%	40.13%
ML-50	6021	2909	214342	35.6	73.68	1.22%	29.54%
Foursquare	485381	83999	1021966	2.10	12.16	0.01%	73.92%
Adressa	141933	3257	1861901	13.12	571.66	0.40%	85.60%

Models: We consider two sequential recommender system models that have been shown to perform well [Lonjarret et al., 2021]:

- Self-Attentive Sequential Recommendation (**SASRec**) [Kang and McAuley, 2018], a self-attention based model that captures both user preferences and user sequential dynamics ;
- Convolutional Sequence Embedding Recommendation (**CASER**) [Tang and Wang, 2018], a CNN-based method that captures both user preferences and user sequential dynamics;

For both models, we used the first 80% of users' actions (Items) in each user's sequence as the training set and the next remaining 20% as the test set for assessing the model's performance. We did not need a validation set since we adopted the hyperparameters used in [Lonjarret et al., 2021]. Indeed, the training and internal optimization processes appear to be the same. Table 6 reports the performance of the models on the benchmarks according to several metrics such as nDCG@K, MAP, Hit@K, Precision@K, and Recall@K (see Appendix A). We observe that the models perform very well. However, their performance can be evaluated by different measures which can be divergent from one model to another.

Aims: Through this empirical study, our goal is to evaluate our subgroup discovery approach to explain the Top-N recommendations. Mainly, we investigate the different functions we use in the quality measure and the two description languages we consider. These experiments aim to bring answers to the following questions:

RQ1 Do the descriptions of the subgroups obtained constitute reliable explanations of the recommendations?
RQ2 Which subgroup definition leads to the best explanations?
RQ3 Are the two languages complementary?
RQ4 Can the explanations alone be used as a recommender system that mimics the studied one?

For each dataset, we consider a random sample of 1400 users whose Top-N recommendations are explained using the proposed approach.

	Metrics	Adressa	Foursquare	ML-50	Video games	Avg
SASRec	AUC	93.87%	94.88%	81.17%	82.32%	88.06%
	MAP	0.23%	1.53%	1.11%	0.131%	0.75%
	NDCG@5	2.66%	16.35%	2.96%	0.96%	5.73%
	NDCG@10	3.58%	19.99%	2.97%	1.30%	6.96%
	HIT@5	3.93%	22.78%	3.51%	1.47%	7.92%
	HIT@10	6.73%	33.75%	4.54%	2.50%	11.88%
	PREC@5	0.03%	0.80%	0.79%	0.03%	0.41%
	PREC@10	0.04%	0.52%	0.71%	0.04%	0.32%
	RECALL@5	0.05%	1.95%	0.62%	0.07%	0.67%
	RECALL@10	0.18%	2.54%	1.13%	0.19%	1.01%
CASER	AUC	83.46%	49.95%	77.86%	80.67%	72.98%
	MAP	13.56%	5.77%	6.25%	1.83%	6.85%
	NDCG@5	9.20%	4.72%	4.06%	1.21%	4.79%
	NDCG@10	12.22%	6.60%	4.51%	1.69%	6.25%
	HIT@5	11.28%	5.86%	5.08%	1.49%	5.67%
	HIT@10	19.88%	10.97%	7.55%	2.85%	10.31%
	PREC@5	7.89%	3.12%	5.93%	0.81%	4.44%
	PREC@10	6.37%	2.04%	5.01%	0.70%	3.53%
	RECALL@5	15.46%	7.89%	5.52%	1.97%	7.71%
	RECALL@10	24.39%	13.0%	9.11%	3.35%	12.46%

Table 6: Performance of the two models on the 4 datasets (Top-5 recommendations).

4.2 General statistics of explanations

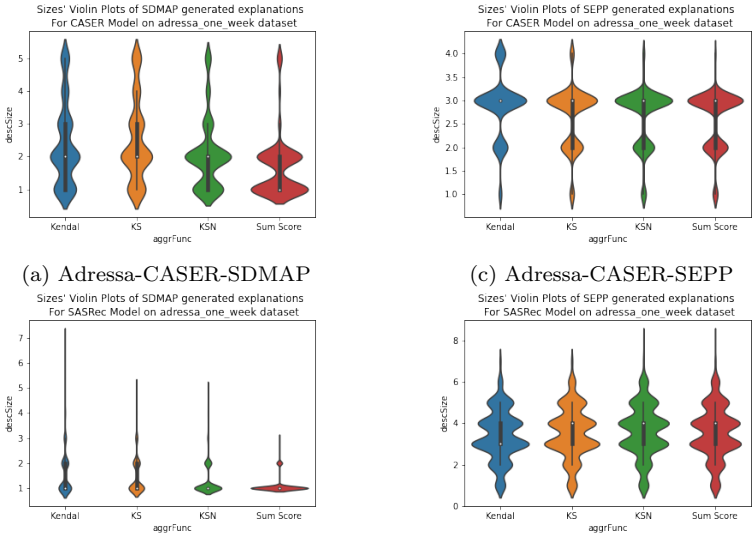
Figure 2 shows the sizes distributions of the explanation generated on the Adressa dataset for both CASER and SASRec models. We can observe that the distributions of the patterns found by SEPP are less sparse than those of SD-Map; this is mainly due to the types of extracted pattern languages we use: sequential patterns in the case of SEPP, which requires at least two items to be fully considered as a sequence, and a relevant set of items for SD-Map, which tends to capture a restricted set of items for better coverage.

To evaluate the quality of the generated explanations, we first report in Table 7 the number of users for whom each method can provide an explanation. Regardless of the policies followed (e.g., Kendall, *KScore*, *KSBounded*, *SumScore*), the number of users explained is always more critical for SD-Map than SEPP. Indeed, for some users, the sequential aspect does not affect the recommendation and only the presence of an item in the user's history is important, whatever its position in it. Therefore, one cannot provide a sequence of two or more items as an explanation, while a set of items can serve as an explanation.

In order to assess the quality of the explanations provided by each method (E^u = SD-Map(TN^u, US) or E^u = SEPP(TN^u, US)), we count the number of users whose sum of scores on the Top-N items increases when only the items of the explanation are provided to the recommendation system. This is formalized as follows:

$$\text{Improved}^\star = \frac{1}{|U|} \sum_{u \in U} \delta_{\sum_{i^\star \in TN^u} r_{i^\star}^{E_u} > \sum_{i^\star \in TN^u} r_{i^\star}^{u}} \tag{6}$$

Table 8 reports the results. On average, the explanations reinforce the model's recommendations for at least 23% and up to 41% of the users. Nevertheless, please note that based on its definition, Improved* favors SumScore policy.



(a) Adressa-CASER-SDMAP

(c) Adressa-CASER-SEPP

(b) Adressa-SASRec-SDMAP

(d) Adressa-SASRec-SEPP

Fig. 2: Distributions of explanation sizes as violin plots of the generated explanations by SD-Map (Resp. SEPP) on recommendations made by CASER in subfigure (a) (Resp. subfigure(c)), and SASRec in subfigure(b) (Resp. subfigure(d)) on the Adressa dataset.

		SD-Map				**SEPP**			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	1377	1384	1379	1377	1268	1124	1268	1268
	Foursquare	1371	1400	1376	1371	1208	911	1208	1208
	ML-50	1066	1066	1066	1066	1066	1063	1066	1066
	Video games	1400	1400	1400	1400	1300	1260	1300	1300
CASER	Adressa	1400	1400	1400	1400	1300	1297	1300	1300
	Foursquare	1208	1208	1208	1208	1108	1108	1108	1108
	ML-50	1066	1066	1066	1066	1066	1060	1066	1066
	Video games	1400	1400	1400	1400	1300	1288	1300	1300

Table 7: Number of users explained by the SD-Map and SEPP algorithms on a sample size of 1400 users (Top-5 recommendations).

Next, we compare the two methods by studying the sum of the scores obtained using SD-Map and SEPP for each policy. More precisely, we count the number of times the method x outperforms the method y in returning a higher score, as expressed by equation 7:

$$\text{Improved}(x, y) = \sum_{u \in U} \delta_{\sum_{i^\star \in TN^u} r_{i^\star}^{E_u^x} > \sum_{i^\star \in TN^u} r_{i^\star}^{E_u^y}} \tag{7}$$

Table 9 shows the respective results of Improved(SD-Map, SEPP) (left) and Improved(SEPP, SD-Map) (right). In most of the cases, SD-Map obtains better scores than SEPP. One can observe that the tendency is different on the Adressa dataset.

		SD-Map				**SEPP**			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	44.71%	22.36%	28.71%	38.29%	**47.00**%	31.71%	38.86%	45.86%
	Foursquare	**56.21**%	30.29%	36.36%	51.57%	45.36%	21.21%	35.29%	44.29%
	ML-50	**50.29**%	24.21%	29.64%	40.50%	38.00%	17.43%	19.36%	33.07%
	Video games	**60.93**%	25.36%	31.29%	48.14%	46.36%	30.29%	33.36%	43.29%
CASER	Adressa	18.07%	15.93%	20.71%	21.29%	23.93%	16.21%	18.21%	**25.71**%
	Foursquare	19.71%	11.64%	16.71%	**21.64**%	16.57%	13.29%	16.79%	18.57%
	ML-50	53.36%	42.14%	50.86%	**57.14**%	43.14%	40.57%	41.57%	42.93%
	Video games	23.86%	13.50%	22.07%	**28.64**%	10.14%	12.36%	12.71%	11.07%
Average		**40.89%**	23.18%	29.54%	38.40%	33.81%	22.88%	27.02%	33.10%

Table 8: Percentage of users whose sum of Top-N recommendation scores (Improved*) increases where only considering the items of the descriptions generated by the SD-Map or SEPP (sample size of 1400 users, Top-5).

These first experiments indicate that SD-Map is better than SEPP on a global view. However, there is not complete domination of SD-Map: For each dataset, at least 25% of users score better with SEPP. Thus, it encourages us to keep both methods in the rest of the experiments. Furthermore, it is important to note that improving the score of the top-N items thanks to the descriptions generated does provide guarantee on the recommendation. The Improved* measure only focuses on the initially recommended Top-N items and monitors the improvement of their individual score. Even if one provides some description that boost the score of the items, the same description may also promote other items that could be integrated to the top-N items. Therefore, we have to take into account this fact to assess the explanation: the explanation we provide has to keep the ranking of the initial Top-N items. This is the objective of the next subsection.

		Number of improved users' scores			
Models	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	**883** , 494	**888** , 496	**847** , 532	**860** , 517
	Foursquare	**847** , 524	**994** , 406	**855** , 521	**852** , 519
	ML-50	**669** , 397	**551** , 515	**587** , 479	**623** , 443
	Video games	**783** , 617	**731** , 669	**724**, 676	**760** , 640
CASER	Adressa	554 , **846**	673 , **727**	**718** , 682	596 , **804**
	Foursquare	**728** , 480	**664** , 544	**758** , 450	**778** , 430
	ML-50	**981** , 85	**597** , 469	**700** , 366	**917** , 149
	Video games	**932** , 468	**758** , 642	**937** , 463	**979** , 421

Table 9: Number of improved users' scores (Improved(SD-Map, SEPP), Improved(SEPP, SD-Map), Top-5 recommendations).

4.3 Assessing the explanations

Evaluating the accuracy of an explanation is difficult due to the lack of ground truth: Even if we know the items that the user consumed, we do not know the reasons that led to this choice. Hence, in the following, we consider G-free metrics to assess the explanations. First, we opt for *Fidelity* [Pope et al., 2019], defined as the difference in accuracy between the recommendations based on the original user history and that obtained by hiding the part of the history corresponding to the

explanation:

$$\text{Fidelity} = \frac{1}{|U|} \sum_{u \in U} 1 - \frac{|t^u \cap t^{u \setminus E_u}|}{N} \in [0; 1]$$

with

- U a sample of user's sequences
- t^u the Top-N items returned by the model when considering the sequence u
- $t^{u \setminus E_u}$ be the Top-N items recommended by considering the sequence u from which the items of the explanation E_u have been removed.

Similarly, we can study the changes of the recommendation by keeping only the essential items (that of the explanation) and by deleting the others, as does the measure of Infidelity:

$$\text{Infidelity} = \frac{1}{|U|} \sum_{u \in U} 1 - \frac{|t^u \cap t^{E_u}|}{N} \in [0; 1]$$

with t^{E_u} the Top-N items recommended by the model when considering the sequence E_u as input. The higher the Fidelity and the lower the Infidelity, the better the explanation. Obviously, masking all the user history would have a significant impact on the model recommendations. Therefore, the former measures should not be studied without considering the *Sparsity* metric that aims to measure the fraction of user sequence selected as an explanation:

$$\text{Sparsity} = \frac{1}{|U|} \sum_{u \in U} (1 - \frac{|E_u|}{|u|})$$

Based on these measures, the best explainable method would achieve high Fidelity, low Infidelity with a sparsity close to 1.

		SD-Map				**SEPP (Support=5)**			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	0.336	0.295	0.319	0.329	0.371	0.365	**0.405**	0.393
	Foursquare	0.187	0.173	0.175	0.186	0.195	0.164	**0.202**	0.197
	ML-50	0.228	0.267	0.273	0.264	0.271	**0.294**	0.290	0.274
	Video games	0.317	0.308	0.316	0.332	0.306	0.333	**0.337**	0.316
CASER	Adressa	0.367	0.363	0.398	**0.401**	0.319	0.326	0.328	0.331
	Foursquare	0.492	0.445	**0.495**	0.490	0.477	0.491	**0.495**	0.487
	ML-50	0.216	0.356	0.357	0.266	0.361	0.365	**0.367**	0.357
	Video games	0.634	0.534	0.605	**0.636**	0.577	0.569	0.578	0.582
Average		0.347	0.343	0.367	0.363	0.360	0.363	**0.375**	0.367

Table 10: Fidelity measures (Top-5 recommendations), the higher the better.

Tables 10, 11, and 12 respectively report the Fidelity, Infidelity, and sparsity scores of the obtained explanations with the SD-Map and SEPP algorithms using the aggregation policies defined in Section 3.3. We can observe that SD-Map does well at extracting faithful explanations by highlighting important sparse explanations, particularly by using *KScore* policy as an aggregation function. On the other hand, we observe that SEPP achieves better results by keeping higher Fidelity while SD-Map offers better Infidelity by selecting explanations that keep the recommended items consistent.

		SD-Map				SEPP (Support=5)			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	0.2767	0.2384	0.2334	**0.2313**	0.2639	0.3451	0.2457	0.2541
	Foursquare	0.2196	0.2279	**0.2177**	0.2229	0.2684	0.4511	0.2599	0.2650
	ML50	0.5683	0.5584	0.5389	0.5557	0.4737	0.4511	**0.4473**	0.4580
	Video games	0.4067	0.3744	0.3706	0.4109	0.3459	0.3331	**0.3071**	0.3363
CASER	Adressa	0.2920	0.2443	**0.2280**	0.2453	0.3054	0.3296	0.2956	0.2956
	Foursquare	0.6816	0.6559	0.6459	0.6293	0.6474	0.6147	**0.5946**	0.6156
	ML50	0.5303	**0.5113**	0.6146	0.6921	0.5641	0.5567	0.5513	0.5616
	Video games	0.5321	**0.4207**	0.4333	0.5091	0.5806	0.5876	0.5719	0.5726
Average		0.437	0.438	**0.404**	0.410	0.431	0.459	0.409	0.420

Table 11: Infidelity measure(Top-5 recommendations), the lower the better.

		SD-Map				SEPP (Support=5)			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	**82.06**%	73.84%	74.78%	79.37%	58.89%	59.94%	58.94%	58.23%
	Foursquare	**77.59**%	73.91%	72.83%	75.02%	55.30%	55.38%	54.71%	55.03%
	ML50	**83.37**%	78.90%	78.72%	80.70%	60.41%	56.69%	56.80%	59.41%
	Video games	**79.00**%	69.58%	69.22%	74.06%	61.29%	58.82%	58.49%	60.62%
CASER	Adressa	**67.77**%	58.18%	55.48%	61.21%	51.67%	48.16%	50.84%	51.09%
	Foursquare	**57.86**%	49.13%	41.13%	47.45%	48.65%	45.38%	45.46%	47.00%
	ML50	**76.46**%	65.23%	65.40%	71.97%	55.36%	54.46%	53.92%	55.00%
	Video games	**63.86**%	48.53%	42.58%	52.15%	51.12%	49.87%	49.79%	50.64%
Average		**73.50**%	64.66%	62.52%	67.74%	55.34%	53.59%	53.62%	54.63%

Table 12: Sparsity measure of SD-Map and SEPP results on SASRec and CASER recommendation systems, where higher values (**in bold**) indicate the explanations are more sparse and tend to only capture the most important input information (Top-5 recommendations).

Accordingly, we propose in the following section to combine our two main approaches, ensuring that the approach is applicable to a wide range of data and that the explanations provided are as simple as possible.

4.4 Impact of method's parameters on the explanation quality

To evaluate the impact of the maximal sequence size on the method, we report on Fig 3 the Fidelity and Infidelity measures when the sizes of the generated sequences $US_1(u)$ and $US_2(u)$ vary. We consider the two recommender systems, SASRec and CASER, but only ML-50 dataset as it has an average number of items per user over 35, while other datasets are below 13 (see Table 5). We can observe that the neighbor sizes have a low impact on the quality of the explanations. Short sequences have already captured the information necessary for the explanation of the models whether for CASER or SASRec model. Indeed, CASER uses a hyperparameter L to specify the number of items of the user history to be considered. In our experiments this value is set to 5.Thus sequences longer than 5 should not provide additional explanation. This is therefore consistent with the observations made in Fig 3. For SASRec model, we set the maxlen hyperparameter to 50. Nevertheless, even if the model can take into account longer sequences, in practice, the relevant information is contained in the shorter recent sequences.

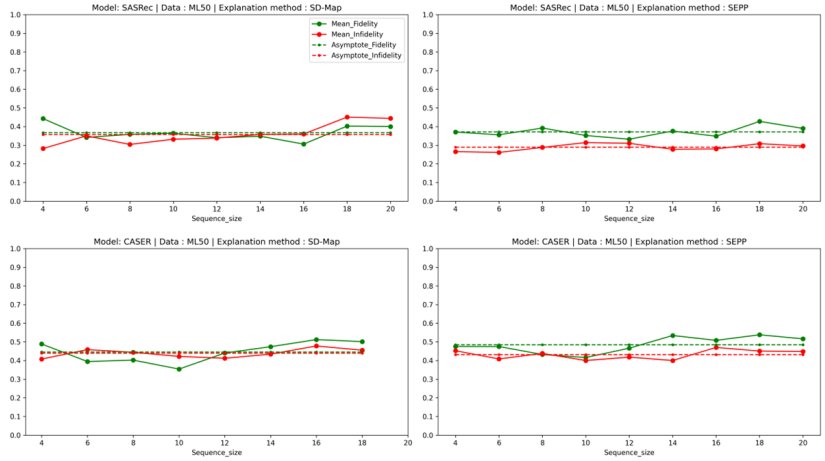


Fig. 3: Evolution of explanation quality measures with various user sequence sizes ($US_1(u)$ at left) for two recommendation models (SASRec (top) and CASER (bottom)). We also add an artificial limitation on the size of sequences of $US_2(u)$ (right) for comparison.

4.5 Are the two languages complementary?

The first observations show that there is not one method that outperforms the other on an entire data set. It is more subtle. There are users for whom SD-Map is more suitable while for others it is SEPP. This motivates us to study how SD-Map and SEPP are complementary. In particular, we want to know what advantages can be derived by combining the two methods. To this end, we consider for each user the explanation that optimizes either Fidelity or Infidelity values:

For Fidelity, we have:

$$E^u = \text{argmax}_{x \in \{E_u^{\text{SD-Map}}, E_u^{\text{SEPP}}\}} \left\{ \left(1 - \frac{t^u \cap t^{u \setminus E_u^x}}{N} \right) \right\}$$

For Infidelity, we have:

$$E^u = \text{argmin}_{x \in \{E_u^{\text{SD-Map}}, E_u^{\text{SEPP}}\}} \left\{ \left(1 - \frac{t^u \cap t^{E_u^x}}{N} \right) \right\}$$

Tables 13 and 14 report the improvement in Fidelity and Infidelity for the combined SD-Map and SEPP methods (Maximum SD-Map and SEPP). We can observe that combining the two methods leads to an average improvement of at least 6.84% in Fidelity and 6.21% for Infidelity. This improvement can go up to 10% according to some configurations. These results also emphasize the efficiency of the *KScore* aggregation function.

		Fidelity Accuracy				**Improvement**			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	0.453	0.436	**0.466**	0.465	6.80%	**9.10%**	6.00%	5.90%
	Foursquare	**0.272**	0.238	0.261	0.268	**7.70%**	6.50%	5.90%	7.10%
	ML50	0.327	**0.36**	0.358	0.343	5.60%	6.60%	6.80%	**6.90%**
	Video games	0.422	0.426	**0.427**	0.425	**10.5%**	9.30%	9.00%	9.30%
CASER	Adressa	0.435	0.454	**0.458**	0.460	6.80%	**9.10%**	6.00%	5.90%
	Foursquare	0.573	0.563	**0.576**	0.570	**8.10%**	7.20%	**8.10%**	8.00%
	ML50	0.390	0.425	**0.430**	0.405	2.90%	6.00%	**6.30%**	4.80%
	Video games	**0.706**	0.655	0.686	0.704	7.20%	**8.60%**	8.10%	6.80%
Average		0.447	0.445	**0.458**	0.455	6.95%	**7.80%**	7.02%	6.84%

Table 13: Max SD-Map-SEPP Fidelity measures (Top-5 recommendations).

		Infidelity Accuracy				**Improvement**			
	Datasets	*SumScore*	*Kendall*	*KScore*	*KSBounded*	*SumScore*	*Kendall*	*KScore*	*KSBounded*
SASRec	Adressa	0.160	0.170	**0.139**	**0.139**	7.10%	**10.7%**	9.90%	7.90%
	Foursquare	0.139	**0.131**	**0.131**	**0.131**	8.40%	8.90%	**10.4%**	8.70%
	ML50	0.437	0.415	**0.411**	0.417	3.70%	3.60%	3.60%	**4.10%**
	Video games	0.250	0.207	**0.190**	0.223	9.60%	**12.6%**	11.7%	1.13%
CASER	Adressa	0.180	0.192	**0.169**	**0.169**	6.50%	**10.0%**	7.50%	5.90%
	Foursquare	0.529	0.521	**0.498**	0.517	**10.0%**	9.40%	9.70%	9.90%
	ML50	0.543	0.464	**0.448**	0.503	2.10%	**6.60%**	6.30%	5.90%
	Video games	0.425	0.417	**0.354**	0.372	8.41%	**11.5%**	6.67%	6.13%
Average		0.333	0.315	**0.292**	0.309	6.98%	**9.16%**	8.22%	6.21%

Table 14: Max SD-Map-SEPP Infidelity measures (Top-5 recommendations).

4.6 Can the explanations alone be used as a recommender system that mimics the studied one?

We experiment with building a global model solely based on the explanations. We first consider the explanations found for each user as a set of rules of the form $E_u \rightarrow TN^u$. We build a global recommender system using a heuristic that is commonly used to solve the weighted set cover problem. More precisely (see Algorithm 1), the greedy method consists of iteratively choosing the explanation that covers the most users and eliminating them for the next iteration. Once the rule cover is computed, we can use it as a Top-N recommendation model. To do this, we retain the N most frequent items among the items appearing in the set of triggered rules by a user.

To measure the performance of this global model, we based our evaluation on the metrics used in Section 4.1. The Results are provided in Figure 4 (SASRec) and Figure 5 (CASER). Notice that we only report the performance of the models based on KScore evaluation of explanations because it gives the best result. On each radar plot, several models are represented: the global model (Algorithm 1) based only on the SD-Map rules in red, the one using the SEPP rules in blue, and in yellow the mixed model using the two types of rules. The perimeter with the white vertices indicates the values of the original model.

We can observe that the rule-based models achieve performances similar to the ones of CASER and SASRec, and even outperform them in some cases (on Adressa and Foursquare for CASER, on Foursquare for SASRec). Regarding the mimicked models, we notice that the SEPP-based global model outperforms the others (SD-Map-based model, mixed-model) while imitating the SASRec recommendation model. Meanwhile, for CASER, it appears that the combination of both SEPP and SD-Map-based explanations allows to build a better model that performs similarly

Algorithm 1 Global model: Explanations as recommender system.

Require: U the set of user sequences, $E = \{E_u \rightarrow TN^u,\ u \in U\}$ the set of explanations, T the type of explanations (either SD-Map or SEPP).
Ensure: R: the set of rules used to recommend items and the Top-N recommended items for a user u.

```
function SELECT_EXPLANATIONS(U, E, T)
    S ← U
    while S ≠ ∅ do
        if T=SD-Map then
            e ← argmax_{(X_e→Y_e)∈E} |{u ∈ S | X_e ∈ I^u}|
            S' → {u ∈ S | X_e ∈ I^u}
        else
            e ← argmax_{(X_e→Y_e)∈E} |{u ∈ S | X_e ⊑ u}|
            S' → {u ∈ S | X_e ⊑ u}
        end if
        R ← R ∪ e
        S ← X \ S'
    end while
    return R
end function
function RECOMMENDER_SYSTEM(u, R, T)
    if T = SD-Map then
        R_u ← {r ≡ X_r → Y_r, r ∈ R | X_r ∈ I^u}
    else
        R_u ← {r ≡ X_r → Y_r, r ∈ R | X_r ⊑ u}
    end if
    Y_u ← {(y, f) | y ∈ Y_r, with X_r → Y_r ∈ R_u, and f = |r ∈ R_u, y ∈ Y_r|}
    return Top-N frequent items in Y_u
end function
```

to the original model or even better. One also can observe that the mixed model outperforms SD-Map-based model in most of the cases.

Table 15 reports the compression ratio (CR) resulting from the different global models on each dataset. It reflects how well we manage to imitate the original model with a relatively small and meaningful set of explanations (i.e. rules). The higher the ratio the better. Also, one of the benefits worth mentioning of the mixed global model is that it increases the compression rate by decreasing the number of explanations on which it is based.

These experiments demonstrate the high quality of the explanations since it is possible to build powerful global models based on them. These global models reach similar performance or outperform the models they mimic. Furthermore, they are small enough to be human-understandable and interpretable.

4.7 Examples

Figure 6 shows specific examples of explanations of SASRec Top-N recommendations. To show how our two explanatory approaches work under varied conditions – whether or not the model succeeds in identifying the ground truth – we have carefully chosen the examples with objective characteristics. The first example is the one with a historic length between 2 and 6 items that maximizes the NDCG@5 value. Hence, SASRec performs well on this example and recommends SUPER MARIO 3D LAND (NINTENDO 3DS game), a similar game to the ground truth

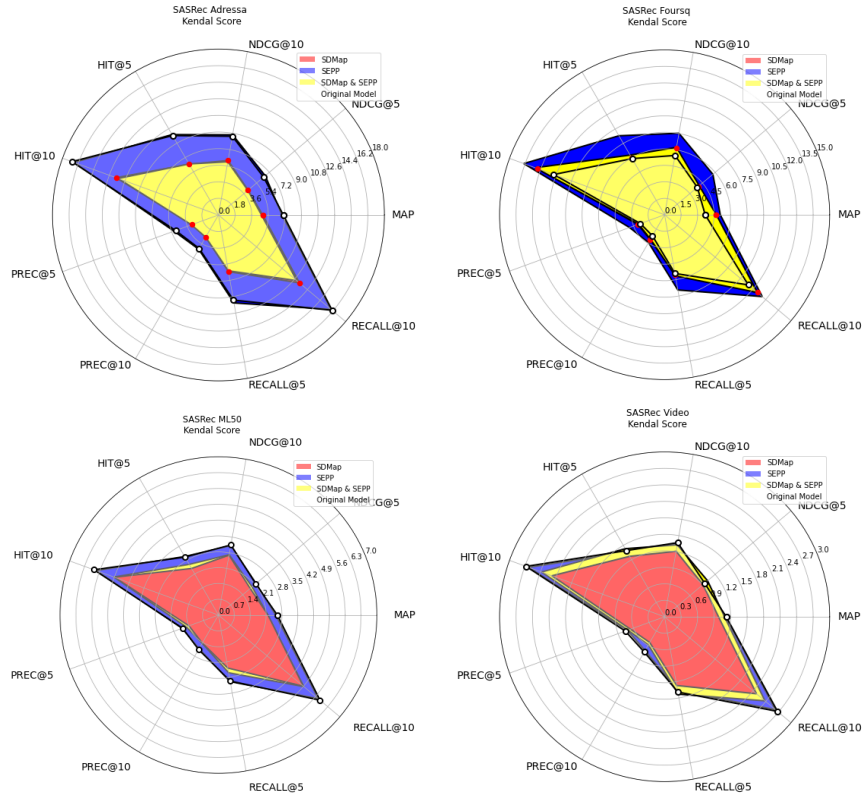


Fig. 4: Radar plot of the performance values (Based on : AUC, MAP, NDCG@k, HIT@k, Precision@k and Recall@k metrics) of the global models based on SASRec on Adressa (top left), Foursquare (top right), ML50 (bottom left) and Video games (bottom right).

one SUPER MARIO BROS. It also recommends three (03) video games related to NINTENDO's consoles (3DS and Wii) and two (02) other items related to XBOX 360 console. The explanations found by SD-Map are made of the last purchased NINTENDO game by the user. SEPP found a sequence whose first item is related to XBOX 360 and the two others are related to NINTENDO video games.

The second example is one where SASRec fails to identify ground truth items. It has been chosen as the one that minimizes NDCG@5 among the examples with length between 2 and 6. SASRec mainly recommends PlayStation 3 hardware and video games. SD-Map explains these recommendations by isolating the two PlayStation 3 video games. SEPP identifies that the user is interested in motoring video games and that recently he turns to PlayStation. This shows that our methods explain the recommendations made by SASRec and not the ground truth.

The third example is the one with maximum mean score on the Top-5 recommended items. SASRec recommends PlayStation 3 war video games (*Call of Duty: Black Ops*) and a joystick for this game console. SD-Map selects war video games for PlayStation 3 among which a previous edition of *Call Of Duty (World At War)*.

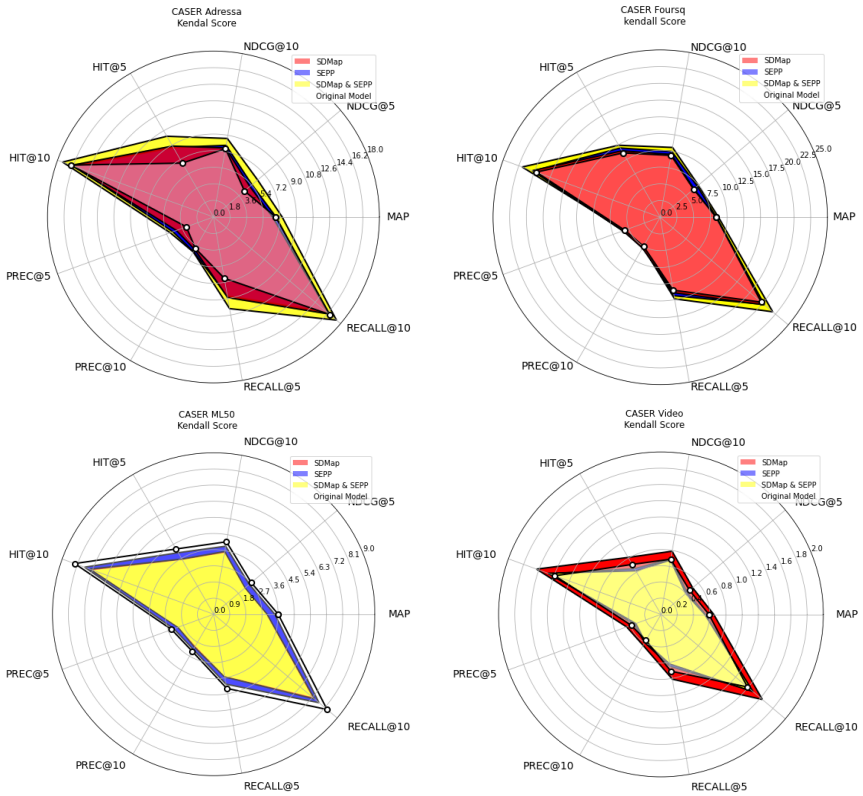


Fig. 5: Radar plot of the performance values (Based on : AUC, MAP, NDCG@k, HIT@k, Precision@k and Recall@k metrics) of the global models based on CASER on Adressa (top left), Foursquare (top right), ML50 (bottom left) and Video games (bottom right).

SEPP identifies a sequence of games often bought together that includes an edition of *Uncharted*, a video game that appears in the Top-N recommendation.

The fourth example is the one that minimizes the Infidelity among those that maximize the Fidelity. We can notice that the items selected by both methods are nearly identical. They highlight role-playing (*Final Fantasy VII*) and fighting games (*Mortal Kombat*) that appear to be an explanation of the Top-1 item (*The Legend of Zelda: Ocarina of Time*), an adventure-style game.

The last example is the one with the best precision@5 and recall@5. Although SASRec got a recall of 1 (perfect recommendation). SD-Map provides *FINAL FANTASY IX* as explanation, that is relevant as 3 of the Top-5 recommendations are related to this license. SEPP explains the recommendation by a sequence of adventure-type games including two chapters (versions) of *ZELDA* game. Looking closely at the latter, we can notice complementarity between the explanations as SD-Map captured the name of the game and SEPP the user's tendency (i.e dynamic behavior) to play successive versions of the same game.

RS	Approach	Policy	Adressa	Foursq	ML50	Video games
SASRec	SD-Map	*SumScore*	92.74%	80.45%	92.21%	54.07%
		Kendall	92.41%	81.50%	91.93%	41.71%
		KScore	92.46%	81.25%	91.65%	42.64%
		KSBounded	92.45%	80.67%	91.84%	47.14%
	SEPP	*SumScore*	91.64%	69.21%	9.47%	13.92%
		Kendall	91.19%	70.25%	10.72%	14.05%
		KScore	91.88%	69.62%	9.57%	14.08%
		KSBounded	91.64%	69.12%	8.54%	13.92%
	SD-Map & SEPP	*SumScore*	96.22%	89.84%	**96.01**%	**79.26**%
		Kendall	95.81%	88.84%	95.96%	73.68%
		KScore	96.07%	**90.21**%	95.78%	74.41%
		KSBounded	**96.22**%	89.92%	95.92%	76.19%
CASER	SD-Map	*SumScore*	93.64%	66.72%	**77.30%**	24.21%
		Kendall	95.00%	72.19%	43.25%	30.71%
		KScore	94.50%	58.61%	49.44%	19.64%
		KSBounded	93.86%	58.20%	66.89%	18.29%
	SEPP	*SumScore*	94.46%	64.62%	15.48%	18.00%
		Kendall	90.13%	67.96%	14.53%	8.00%
		KScore	94.69%	65.79%	11.63%	15.69%
		KSBounded	95.08%	66.79%	14.73%	17.85%
	SD-Map & SEPP	*SumScore*	97.22%	86.96%	71.67%	60.44%
		Kendall	**97.52**%	**87.95**%	71.45%	**60.68**%
		KScore	97.41%	85.54%	71.20%	58.33%
		KSBounded	97.30%	85.92%	68.25%	58.59%

Table 15: Compression rate (CR) of the global models that mimic a recommendation system; CR is the proportion reduction in the number of possible explanations defined as (1 - #E'/#E) where #E and #E' are, respectively, the number of unique explanation before and after constructing the global model. In the case of the mixed global model #E is equal to the sum of both SD-Map and SEPP explanation (#E=#E(SD-Map)+#E(SEPP)). The best approach and policy for each recommender system and each dataset is highlighted.

5 Conclusion

In this paper, we presented a method for explaining the Top-N recommendations made by a recommender system. We used the recommendation history utilizing subgroup discovery techniques to identify the operational data used for recommendation. In general, this methodological approach to generating explanations based on histories is independent of the model and has broad applicability.

We proposed to use two pattern languages, one time-agnostic, the other one time-sensitive and studied four aggregation policies of the Top-N recommendation values, i. e., the set of items.

For evaluating and assessing the proposed approach, we presented a set of experiments performing a deep analysis of 2 state-of-the-art models constructed on 4 datasets. We presented an extensive discussion of our results in context. These experiments reveal that the proposed approach provides local explanations based on the user preferences (SD-Map) or on the order of actions performed by the user (SEPP) if this order impacts the recommendation. We cannot conclude that SD-Map provides better results than SEPP. Overall, SD-Map performs better than SEPP with regard to several indicators (i. e., Improved*) but this only means that local recommendations are explainable based on the user-preferences only for a large proportion of users (RQ1 and RQ2). Interestingly, SEPP is complementary and

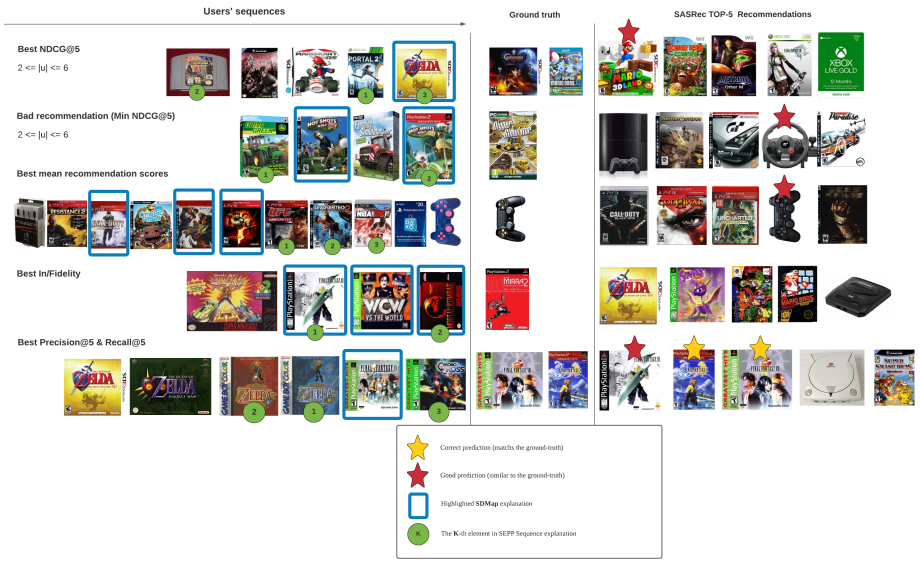


Fig. 6: Explanation examples found in Video games dataset. Yellow stars highlight accuracy in the recommendations, whereas red stars identify "good" recommendations (i.e., the related items are similar to the ground truth).

allows to improve the explanation reliability for users whose sequential dynamics is a key factor of the recommendation. Further tests show that simultaneously considering both SD-Map and SEPP leads to better explanation in term of fidelity and infidelity with respectively average improvements of at least 6.84% and 6.21% (RQ3). Our results also suggest that the *KScore* aggregation function is the most promising function among the four functions we considered. Eventually, our experiments highlight that it is also possible to provide global explanations of a model based on a wisely selected set of local explanations. The identified explanations are sufficiently precise to mimic the original recommender systems and to be used as a model themselves (RQ4).

We believe that SD-Map and SEPP can support the explanations of Top-N recommendation made by any recommender system that takes as input a user history only. However, a number of potential limitations need to be considered for future research to make these methods fully effective in practice for any type of models.

First, our approach remains costly, especially if we want to explain a large number of users. To overcome this limitation, the completeness must be relaxed and some heuristic-based algorithms used (e.g., Beam-search or Monte-Carlo Tree Search-based algorithms). Furthermore, we provide, in this paper, explanations for each user in an independent way. We can imagine more global process to handle the high number of users to explain (e.g., group users with similar recommendations and/or with similar action histories).

Experiments suggest that for most of the users, SD-Map (i. e., user preferences) is enough to explain the Top-N recommendations while SEPP (i. e., sequential

dynamics) is the complementary method to consider for a non negligible part of users. Our empirical study did not reveal why for some users the sequential aspect does not affect the recommendation. This absence of justification is a limitation which requires to investigate the item embedding built by the model to be overcame. Further studies, which aims at characterizing how the items' embedding is built based on user preferences and the sequential dynamics, will need to be undertaken.

Eventually, SEPP and SD-Map make it possible to explain the Top-N recommendations based only on the past user's actions. Nevertheless, we believe that these two methods can be extended to be applicable to any recommendation system. Further interesting options concern exploiting additional information about the time and the respective attributes of the item interactions as well as the user information.

Acknowledgments

This work was supported by the ACADEMICS grant of the IDEXLYON project of the University of Lyon, PIA operated by ANR-16-IDEX-0005. It also benefited from the financial support CAF AMERICA OPE2020-0041.

A Appendix

Hit rate (Hit@N):

$$Hit@N = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in GT^u} \frac{\mathbb{1}(R_i \leq N)}{|GT^u|}.$$

where the indicator function $\mathbb{1}(b)$ returns 1 if its argument b is *True*, *0* otherwise. R_i is the ranking of the ground-truth item i. The HIT@N function returns the average number of times the ground-truth item is ranked in the Top-N items. We compute HIT@5, HIT@10, HIT@25.

Normalized Discounted Cumulative Gain at position N (nDCG@N):

$$nDCG@N = \frac{1}{|U| \times N} \sum_{u \in U} \sum_{i_t \in GT^u} \frac{\mathbb{1}(R_{i_t} \leq N)}{log_2(\max(R_{i_t} + 2 - t), 2)},$$

The NDCG@k is a position-aware metric which assigns larger weights to higher positions. We compute NDCG@5, NDCG@10, NDCG@25 and NDCG@50.

Area Under Curve (AUC):

$$AUC = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|GT^u|} \sum_{i_t \in GT^u} \frac{|I| - R_{i_t} - t}{|I|}.$$

This measure calculates how high the ground-truth items of each user has been ranked in average.

Precision at N (Precision@N):

$$Precision@N = \frac{1}{U} \sum_{u \in U} \frac{TN^u \cap GT^u}{N}$$

Recall at N (Recall@N):

$$Recall@N = \frac{1}{U} \sum_{u \in U} \frac{TN^u \cap GT^u}{|GT^u|}$$

Mean Average Precision (MAP):

$$MAP = \frac{1}{|U|} \sum_{u \in U} \frac{\sum_{k=1}^{N} Precision@k \times rel(k)}{N}$$

where $rel(k) = 1$ if the k-th item in TN belongs to GT, the ground Truth items of the user.

References

Atzmueller, 2015. Atzmueller, M. (2015). Subgroup Discovery. *WIREs Data Mining and Knowledge Discovery*, 5(1):35–49.

Atzmueller and Lemmerich, 2009. Atzmueller, M. and Lemmerich, F. (2009). Fast Subgroup Discovery for Continuous Target Concepts. In *Proc. International Symposium on Methodologies for Intelligent Systems*, volume 5722 of *LNCS*, pages 1–15, Berlin/Heidelberg, Germany. Springer.

Atzmueller and Puppe, 2006. Atzmueller, M. and Puppe, F. (2006). SD-Map - A Fast Algorithm for Exhaustive Subgroup Discovery. In *Proc. PKDD*, pages 6–17. Springer.

Falher et al., 2015. Falher, G. L., Gionis, A., and Mathioudakis, M. (2015). Where is the soho of rome? measures and algorithms for finding similar neighborhoods in cities. In *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*, pages 228–237.

Fournier-Viger et al., 2017. Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1):54–77.

Fürnkranz et al., 2020. Fürnkranz, J., Kliegr, T., and Paulheim, H. (2020). On cognitive preferences and the plausibility of rule-based models. *Mach. Learn.*, 109(4):853–898.

Gulla et al., 2017. Gulla, J. A., Zhang, L., Liu, P., Özgöbek, O., and Su, X. (2017). The adressa dataset for news recommendation. In *Proceedings of the International Conference on Web Intelligence*, WI '17, page 1042–1048, New York, NY, USA. Association for Computing Machinery.

Han et al., 2000. Han, J., Pei, J., and Yin, Y. (2000). Mining Frequent Patterns Without Candidate Generation. In *Proc. ACM SIGMOD Intl. Conference on Management of Data*, pages 1–12. ACM Press.

Harper and Konstan, 2015. Harper, F. M. and Konstan, J. A. (2015). The movielens datasets: History and context. *ACM TiiS*, 5(4):19:1–19:19.

Henelius et al., 2014. Henelius, A., Puolamäki, K., Boström, H., Asker, L., and Papapetrou, P. (2014). A peek into the black box: Exploring classifiers by randomization. *Data Min. Knowl. Disc.*, 28(5-6):1503–1529.

Kang and McAuley, 2018. Kang, W. and McAuley, J. J. (2018). Self-attentive sequential recommendation. In *Proc. ICDM*, pages 197–206. IEEE.

Kendall, 1938. Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Klösgen, 1996. Klösgen, W. (1996). Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. AAAI.

Lemmerich et al., 2016. Lemmerich, F., Atzmueller, M., and Puppe, F. (2016). Fast exhaustive subgroup discovery with numerical target concepts. *Data Min. Knowl. Disc.*, 30(3):711–762.

Lemmerich et al., 2012. Lemmerich, F., Becker, M., and Atzmueller, M. (2012). Generic Pattern Trees for Exhaustive Exceptional Model Mining. In *Proc. ECML/PKDD*, Berlin/Heidelberg, Germany. Springer.

Lonjarret et al., 2021. Lonjarret, C., Auburtin, R., Robardet, C., and Plantevit, M. (2021). Sequential recommendation with metric models based on frequent sequences. *Data Min. Knowl. Discov.*, 35(3):1087–1133.

Lonjarret et al., 2020. Lonjarret, C., Robardet, C., Plantevit, M., Auburtin, R., and Atzmueller, M. (2020). Why should I trust this item? explaining the recommendations of any model. In Webb, G. I., Zhang, Z., Tseng, V. S., Williams, G., Vlachos, M., and Cao, L., editors, *7th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2020, Sydney, Australia, October 6-9, 2020*, pages 526–535. IEEE.

Mabroukeh and Ezeife, 2010. Mabroukeh, N. R. and Ezeife, C. I. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43(1):3:1–3:41.

Mandel, 2007. Mandel, D. R. (2007). Counterfactual and Causal Explanation: From Early Theoretical Views To New Frontiers. In *The Psychology of Counterfactual Thinking*, pages 23–39. Routledge.

Mathonat et al., 2019. Mathonat, R., Nurbakova, D., Boulicaut, J., and Kaytoue, M. (2019). Seqscout: Using a bandit model to discover interesting subgroups in labeled sequences. In *Proc. DSAA*, pages 81–90.

McAuley et al., 2015. McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *Proc. SIGIR*, pages 43–52. ACM.

McSherry, 2005. McSherry, D. (2005). Explanation in recommender systems. *Artificial Intelligence Review*, 24(2):179–197.

Mollenhauer and Atzmueller, 2020. Mollenhauer, D. and Atzmueller, M. (2020). Sequential exceptional pattern discovery using pattern-growth: An extensible framework for interpretable machine learning on sequential data. In Atzmüller, M., Kliegr, T., and Schmid, U., editors, *Proceedings of the First International Workshop on Explainable and Interpretable Machine Learning (XI-ML 2020) co-located with the 43rd German Conference on Artificial Intelligence (KI 2020), Bamberg, Germany, September 21, 2020 (Virtual Workshop)*, volume 2796 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Pei et al., 2001. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns by prefix-projected growth. In Young, D. C., editor, *Proc. International Conference on Data Engineering*, pages 215–224, Los Alamitos. IEEE.

Pei et al., 2004. Pei, J., Han, J., Mortazavi-Asl, B., W., J., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2004). Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424 – 1440.

Pope et al., 2019. Pope, P. E., Kolouri, S., Rostami, M., Martin, C. E., and Hoffmann, H. (2019). Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10772–10781.

Pu and Chen, 2006. Pu, P. and Chen, L. (2006). Trust building with explanation interfaces. In *Proc. IUI*, pages 93–100.

Ribeiro et al., 2016. Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. In *Proc. KDD*, pages 1135–1144.

Ribeiro et al., 2018. Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-Precision Model-Agnostic Explanations. In *Proc. AAAI*.

Roth-Berghofer et al., 2007. Roth-Berghofer, T., Schulz, S., Leake, D., and Bahls, D. (2007). Explanation-aware computing. *AI Magazine*, 28(4).

Roth-Berghofer and Cassens, 2005. Roth-Berghofer, T. R. and Cassens, J. (2005). Mapping Goals and Kinds of Explanations to the Knowledge Containers of Case-Based Reasoning Systems. In *Proc. ICCBR*, number 3620 in LNAI, pages 451–464, Berlin/Heidelberg. Springer.

Schank, 1986. Schank, R. C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Sørmo et al., 2005. Sørmo, F., Cassens, J., and Aamodt, A. (2005). Explanation in case-based reasoning – perspectives and goals. *Artificial Intelligence Review*, 24(2):109–143.

Tang and Wang, 2018. Tang, J. and Wang, K. (2018). Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 565–573, New York, NY, USA. ACM.

Tintarev and Masthoff, 2011. Tintarev, N. and Masthoff, J. (2011). Designing and evaluating explanations for recommender systems. In *Recommender systems handbook*, pages 479–510. Springer.

Wick and Thompson, 1992. Wick, M. R. and Thompson, W. B. (1992). Reconstructive expert system explanation. *Artificial Intelligence*, 54(1-2):33–70.

Wrobel, 1997. Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. In *Proc. PKDD*, number 1263 in LNCS, pages 78–87, Berlin/Heidelberg, Germany. Springer.

Zaki, 2001. Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2):31–60.